Universidad Carlos III de Madrid

CONTROL SYSTEM FOR AUTONOMOUS LANDING ON MOVING PLATFORMS

Trabajo Fin de Grado.

Una disertación para la Universidad Carlos III de Madrid, siguiendo los requisitos del grado de INGENIERÍA ELECTRÓNICA INSDUSTRIAL Y ATOMÁTICA en la Escuela Politécnica Superior de Leganés.

Alejandro Vázquez Ruano

JUNIO 2017

TUTOR DEL PROYECTO: Abdulla Al Kaff

RESUMEN

En los tiempos que vivimos, el auge existente en los UAV es notable. Han ido adquiriendo competencias múltiples debido a su gran fiabilidad, operatividad económica, y capacidad para adaptarse a un gran rango de entornos. Este proyecto se centra en los Quadcoter, que son uno de los UAV's más usados hoy en día, debido a su gran maniobrabilidad y mecánica simple. Aterrizar en cualquier sitio sin necesidad de un piloto ya esté tanto dentro como fuera de la aeronave, se ha convertido en uno de los campos de estudio sobre Drones más amplio y desarrollado.

Este documento muestra el Trabajo Fin de Grado titulado *'CONTROL SYSTEM FOR AUTONOMOUS LANDING ON MOVING PLATFORMS'*, desarrollado en el Laboratorio de Sistemas Inteligentes de la Universidad Carlos III de Madrid, en la Escuela Politécnica superior de Leganés.

En este proyecto se implementará un controlador PID cuya función principal es aproximar al vehículo aéreo a una posición de aterrizaje determinada. El Drone va restando progresivamente los errores de la distancia hasta el destino y el valor se introduce en el controlador, este lo traduce a comandos de Roll Pitch y Throttle, que se mandan a una controladora para implementar una fuerza determinada a cada uno de los cuatro motores del Drone y de esta manera dirigirlo al destino.

El primer experimento realizado para analizar el controlador consiste en posicionar el Drone en un punto inicial en el espacio aleatorio y a su vez una posición de destino (aterrizaje). Y observar que tendencia tienen los comandos de Pitch, Roll y Throttle. Cuanto más alejando esté el Drone del punto final, mayor será el comando devuelto, por lo tanto mayor es la fuerza que se implementa a cada uno de los motores. Los siguientes experimentos consisten en obtener la respuesta del controlador a medida que el Quadcopter recorre una trayectoria. Las trayectorias simuladas van desde una simple línea recta, hasta una aproximación desde una altura 'H', recorriendo 'N' puntos de coordenadas de un espacio de 3D.

Palabras clave: Drone, Controlador, Comando, UAV, Pitch, Roll, Throttle, PID, Autónomo, Trayectoria.

ABSTRACT

Nowadays, UAV's influence is very remarkable. They have been acquiring multiple roles, thanks to their reliability, economical operations and capacity to adapt to a great range of environments. The main topic of this project is focused on Quadcopters. They are one of the most used UAV's because their great maneuverability and simple mechanics. Land in every single place without a pilot either inside or outside the aircraft, has become one of the biggest and most developed Drone's study fields.

In this document it is shown the Final Project Degree named '*CONTROL SYSTEM FOR AUTONOMOUS LANDING ON MOVING PLATFORMS*', developed in the Intelligent System Lab of the Universidad Carlos III de Madrid, in the Superior Technical School of Leganés.

In this project is going to be implemented a PID controller. Which its main role is to approach the aerial vehicle to a landing position. Firstly the Quadcopter is going to subtract the distance error between the actual position and final one thanks to the GPS. This error will be the input of the PID controller, and finally the controller translates it into an effort which is sent to PX4 internal controller to applying a force in each of the four Drone engines.

The first experiment to analyze the PID controller behavior, consists in pre-program a random array of 3D initial positions, also a landing point in order to observe the trend of each command which is the PID output. As farther the Drone is from landing position, greater would be the command returned, as well as the engines power will be greater. The following experiments consist in obtaining the controller response at the same time the Drone is flying through a path. Simulated paths have different elaboration, from a simple straight line complexity to approach from an 'H' altitude, flying through 'N' points of coordinates in 3D space.

Key Words: Drone, Controller, Command, UAV, Pitch, Roll, Throttle, PID, Autonomous, Path.

AGRADECIMIENTOS

Como no puede ser de otra manera, en primer lugar, me gustaría agradecerle a mi tutor de TFG Abdulla Al Kaff el haberme dado la oportunidad de realizar este proyecto y poder iniciar mis conocimientos de ingeniería de control aplicados a un Drone. A su vez me gustaría agradecer al personal del Laboratorio de Sistemas Inteligentes todas las dudas que me han resuelto a lo largo de estos meses. Y a mis amigos Eduardo Martín por su esmero en explicarme el funcionamiento de ROS y a Pablo Agudo por sus lecciones de Ubuntu y de todos los conceptos informáticos que me han ayudado implementar mis programas.

Por otro lado he de agradecerles a mis Padres, María Jesús y José Manuel, Hermana, Abuelos y Tíos, su incondicionalidad por todo el apoyo recibido durante la realización de este Trabajo de Fin de Grado, especialmente a mi Tío Roberto con el que he pasado días y días en la Universidad.

GRACIAS

DECLARACIÓN DEL AUTOR

Declaro que el trabajo en esta disertación ha sido desarrollado de acuerdo a los requerimientos de la Universidad Carlos III de Madrid. Y no se ha copiado información de otros estudios, o proyectos de investigación sin haber realizado una correcta referencia en el texto. El Trabajo Fin de Grado es propiedad de su propio autor. El trabajo realizado en colaboración o la asistencia de otras personas también es indicado. Cualquier punto de vista expresado en la disertación es del Autor.

FIRMADO:  FECHA: 21/06/2017

LISTA DE CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

Capítulo 1

I. INTRODUCCIÓN

Hoy en día existe una gran demanda de vehículos aéreos no tripulados, UAV (Uhnmanned Aerial Vehicle), por sus siglas en inglés. Este auge se debe a la capacidad multitarea que pueden llegar desarrollar de una forma eficiente, económica y segura para el ser humano. Las operaciones que un UAV puede desempeñar son múltiples; como la observación para estudios geográficos, repartidor (Amazon Drone), tareas de salvamento etc. Todas ellas capaces de mejorar la vida de las personas.

Los UAV existentes hoy en día, se pueden dividir principalmente en varios tipos, estos son los de ala fija (Fixed wing UAV), helicópteros (Rotorcraft), y drones propulsados por varios motores eléctricos (Multicopter).

Estos últimos utilizan hélices que giran en sentidos opuestos y simétricos, realizando un empuje generalmente perpendicular a la superficie del suelo, no necesitan estabilizador horizontal, vertical como los aviones de alas fijas, o rotores articulados como los de los helicópteros, ya que las únicas superficies de empuje y sustentación son simples hélices que controlan un empuje determinado y permiten moverse el UAV con los típicos ángulos de navegación; Roll (φ), Pitch (θ) y Yaw (ψ). Los Drones propulsados por varias hélices pueden llegar tener hasta ocho rotores (Octacopter). Pero en este proyecto, trabajaremos con el Quadcopter, que como su nombre indica, tiene cuatro hélices. A su vez los Quadcopters tienen dos configuraciones de movimiento, en 'X' o en '+', como se muestra a continuación:

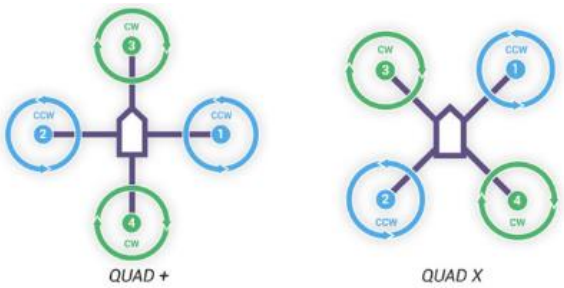

Figura I.1 Configuración de un Quadcopter 'X' y '+ '[32].

Para un análisis en mayor profundidad sobre los tipos de UAV's presentes hoy en día con sus ventajas e inconvenientes se muestra la siguiente tabla [18]:

Tabla I.1 *Tipos de UAV's existentes, ventajas e inconvenientes [18].*

Categoría	Configuración (eje.)	Ventajas	Desventajas
UAV de ala fija	Ala fija (AV)	Mecánica simple, Operaciones silenciosas.	No vuelo de estacionamiento
UAV de Alas Rotativas, Multicopter	Único Rotor (A. V de Rostyne)	Buena controlabilidad, buena maniobrabilidad	Mecánica Compleja, rotores largos y larga cola
	Rotores Axiales (Maryland University)	Mecánica simple. Compacto	Aerodinámica compleja
	Rotores Coaxiales (EPSON)	Mecánica simple. Compacto	Aerodinámica compleja
	Rotores en Tándem (Heudiasyc)	Buen control, aerodinámica simple	Mecánica compleja, tamaño grande
	Quadrotor (EPFL-ETHZ)	Buena maniobrabilidad, mecánica simple, incremento de la carga de pago	Gran consumo de energía
UAV de Alas de aleteo	Movimiento estilo insecto (UC Berkeley)	Buena maniobrabilidad, compacto	Mecánica compleja
	Movimiento estilo pez. (US Naval Lab)	Modos múltiples de movilidad y aerodinámica eficiente	Control complejo, difícil maniobrabilidad
UAV dirigible	Dirigible (EPFL)	Poca energía, largas misiones y modo "auto-lift"	Gran tamaño, difícil control
	Hibrido Quadrotor-Dirigible (MIT)	Buena maniobrabilidad	Gran tamaño, débil al entorno

La parte del proyecto en la que se trabajará es el control del Quadcopter para que pueda realizar un aterrizaje seguro de forma autonomía sobre una plataforma móvil.

Este proyecto ha sido realizado dentro del equipo de investigación formado en el LSI (Laboratorio de Sistemas inteligentes) de la Universidad Carlos III de Madrid.

Motivación

Es muy probable que hayan visto el gran desarrollo del estudio en sobre Multicopter en estos últimos diez años. Esto es debido a que necesitamos investigar y avanzar en el tema, ya que son una nueva forma de vehículos voladores, capaces de acceder a sitios remotos de una manera muy económica y eficiente. El control de estas aeronaves no tripuladas puede que sea una de los campos más innovadores del sector,

ya que tienen que implementarse tareas, despegues, evasión de obstáculos [13, 14] y aterrizajes sin piloto [1-10].

Hay empresas y gobiernos que están desarrollando planes para sumar drones a sus plantillas de trabajo y que hagan funciones de reparto, otras como Amazon Prime Air [15] ya los tienen incorporados, poseen un control autónomo total para realizar envíos de una manera muy rápida, y aterrizan de forma autónoma en un "setpoint" previamente programado. Son capaces de omitir prácticamente todos los obstáculos, y esto es gracias a la programación de su control que va dentro de su ordenador de a bordo. Otra empresa Irlandesa denominada Pony Express [16], fue capaz de lanzar sobre una embarcación una carga de pago urgente, en un tiempo escaso sin necesidad de piloto alguno, esto fue gracias al control programado, aunque por ahora solo se están haciendo envíos de carga barata, poco pesada y urgente debido a la poca autonomía de los Drones hoy en día [17].

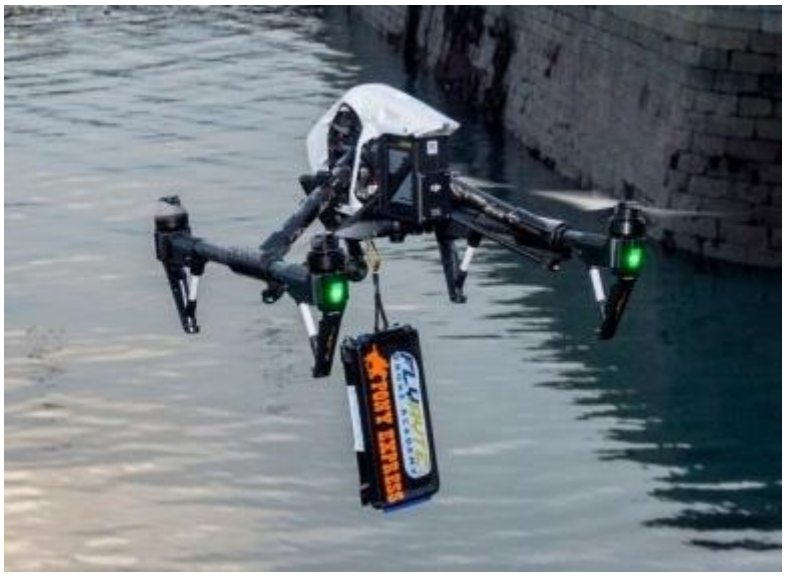

Figura I.2 'Ponny Express' entrega de carga [16].

El control es probablemente la parte menos vistosa de un Multicopter, y la que menos atrae a la gente, pero es una de las más importantes ya que si se implementa correctamente puede habilitar al Drone a hacer cualquier tarea.

Siempre he tenido un gran interés en las aeronaves, y creo que este primer contacto en el tema es una oportunidad que no podía dejar escapar. En definitiva, la motivación principal de este proyecto es una primera toma de contacto en la programación de un control para un Multicopter, con el que voy a tener la oportunidad de incrementar mis

conocimientos en el tema para en un futuro seguir optimizando un proceso de aterrizaje autónomo para aplicárselo a una aeronave para que este pueda facilitar la vida a las personas optimizando el modo de vuelo y sus tareas.

Problema que se nos presenta

El principal dilema de este proyecto es cómo lograr que un Drone Quadcopter en configuración X aterrice sobre una plataforma siguiendo una trayectoria de puntos determinada sin ayuda de un piloto. El proceso sería el siguiente:

El Quadcopter parte de una posición de vuelo de estacionamiento (Hovering), posteriormente los datos procesados de la cámara detectarían el patrón de aterrizaje y un controlador sería el encargado de computar la distancia y aproximar gradualmente el Drone a su destino final. En el momento en el que llega al suelo, los motores se desarman y el resultado sería satisfactorio. Por otro lado puede ocurrir que en el proceso de aterrizaje, el Quadcopter se desvíe, en este caso el Controlador debería de procesar la nueva posición actual y devolver el Drone a una aproximación segura.

Solución que se aporta, el PID Controller

El principal objetivo de esta disertación es realizar el controlador que ayude a aterrizar al Drone correctamente en una posición determinada. Para ello se emplea un controlador Proporcional Derivativo e Integral.

Los controladores que se han barajado para su implementación en el proceso de aterrizaje son el P, PI, PD y PID. Al final nos decantamos por el PID debido a que el controlador P, el más sencillo, tiene una sobreoscilación grande y un error de régimen permanente. El controlador PI no tiene un error de régimen permanente pero tiene una respuesta con una sobreoscilación máxima, la cual puede ser perjudicial en un vuelo estable para el Quadcopter. Por otro lado el Controlador PD tiene una menor sobreoscilación debido a la componente derivativa, pero en este caso el controlador tiene error en régimen permanente.

Al final nos decantamos por el Controlador PID, porque fusiona todas las características anteriores, no tiene error en régimen permanente gracias a la componente Integral, y a su vez la sobreoscilación se puede anular en gran medida con una componente derivativa.

Tabla I.2 Características de los controladores según tipo.

	Controladores			
	P	*PD*	*PI*	*PID*
Error en régimen Permanente	Sí	Sí	No	No
Sobreoscilación	Sí	No	Sí	No

La representación básica de un controlador PID en diagrama de bloques es la siguiente:

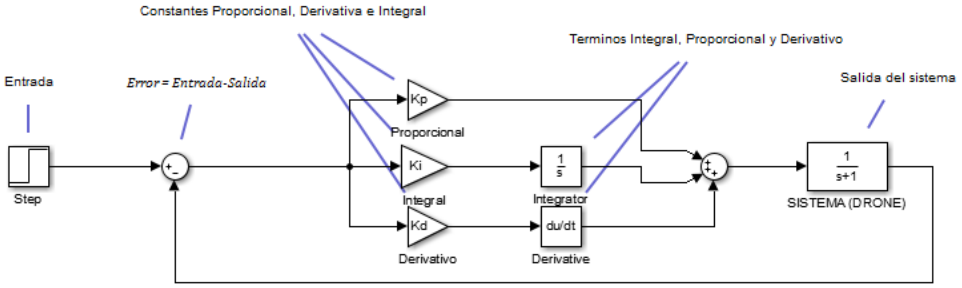

Figura I.3 Diagrama de Bloques de un Controlador PID.

La fórmula matemática del conjunto anterior es:

$$Entrada\ al\ Sistema\ (Drone) = K_P(error) + K_D\frac{d(error(t))}{dt} + K_I\int_0^t error(t)\,dt$$

$$\forall\ error = X_{final} - X_{actual}$$

Ecuación I-1 Ecuación de un controlador PID.

La función básica de un controlador PID es ir restando la posición actual a la de referencia y obtener un error, multiplicar este error por unas constantes determinadas Kp, Ki, Kd, y en el caso de las componentes derivativa e integral, realizar la derivada del error y la integral del error

con respecto el tiempo respectivamente. Esas tres componentes se suman y el valor se transfiere al sistema, en este caso a un Drone. Este valor será el comando que se enviará a la controladora interna del Drone para que transmita más o menos potencia a cada uno de los rotores en una aproximación gradual.

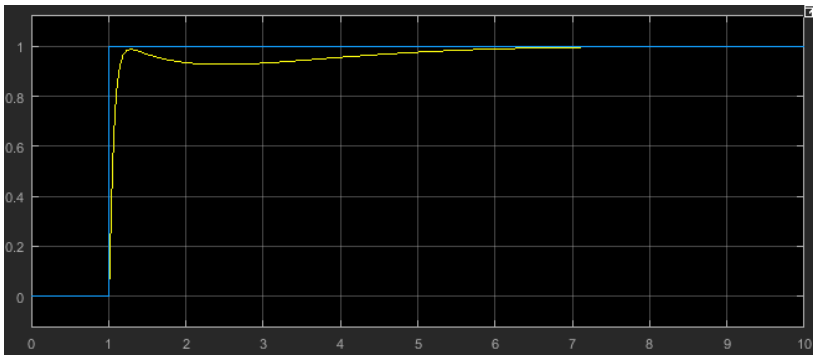

Figura I.4 *Respuesta de un controlador PID antes una entrada escalón.*

En la figura anterior se puede observar la respuesta de un sistema con un controlador PID ante una respuesta escalón, la componente azul representa el destino final del Drone, y la componente amarilla la posición de éste a lo largo del tiempo. Como se puede observar existe en el segundo 1.25 una sobreoscilación, la cual se minimiza con la componente derivativa. A su vez se aprecia que el error en régimen permanente no existe, ya que a partir del segundo 7 el sistema se estabiliza en la posición final.

Debido a que el Quadcopter se traslada en 3 Dimensiones, se ha de aplicar un controlador en cada uno de los ejes de coordenadas: 'X', 'Y', y 'Z'. Para lograr la aproximación en el eje X, se realiza un controlador que envía el comando del ángulo Pitch a la controladora. En la distancia Y el Quadcopter se aproxima variando el ángulo Roll. Y en la disminución de altitud el comando que se ha de ir modificando es el Throttle.

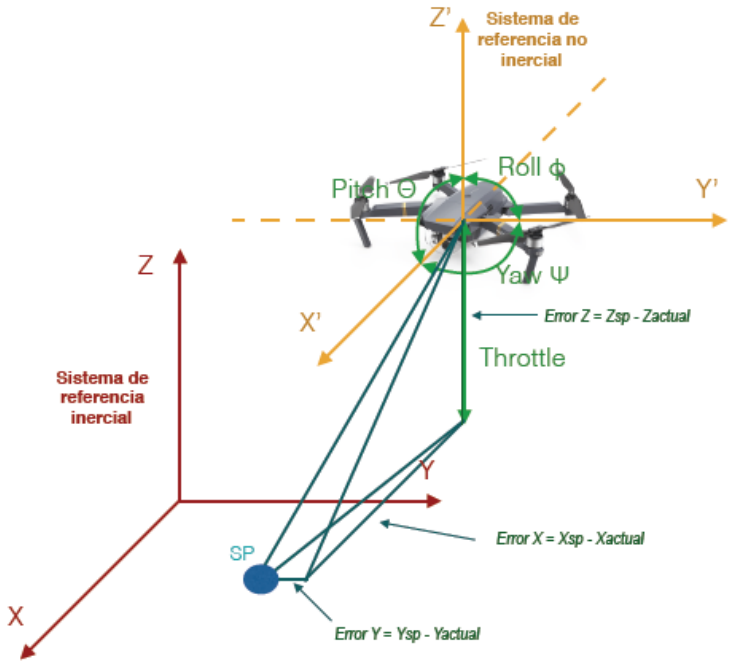

Figura I.5 Movimientos del Drone en los tres ejes [31].

Una vez conocido el tipo y el número de controladores que se van a utilizar, hay que implementarlos. Para ello se ha decido utilizar el lenguaje de programación C++ y Robotic Operating System. Con C++ se crean los ejecutables, y mediante los mensajes Mavlink de MAVROS, un nodo de ROS, se conecta la información recibida del Drone con el controlador PID realizado, y a su vez, se envía la salida del Controlador PID a la controladora del Drone. La controladora que se dispone para este proyecto es una PixHawk PX4.

Estructura del proyecto

Para facilitar la lectura, se mostrará a continuación los contenidos resumidos de cada capítulo que están contenidos en este proyecto.

- **Capítulo 1.** Introducción. Es el capítulo en el que nos encontramos ahora, en primer lugar de contextualizará el proyecto mostrando que es un UAV y una tabla donde aparecen la variedad de UAVs hoy en día. Posteriormente se explicará el motivo que ha impulsado a realizar este proyecto. A continuación se describe cual es el problema que se tiene que abordar y la solución que se aporta. Por último se presenta un pequeño resumen del contenido de cada capítulo del proyecto.

- **Capítulo 2.** Estado de la tecnología. Este apartado muestra una gran variedad de sistemas de control de aterrizaje de UAV's que se

emplean hoy en día. Se explican de uno en uno mostrando figuras, esquemas y ecuaciones que ayuden a su entendimiento. Cada uno de los controladores viene acompañado con una referencia de dónde se ha recuperado la información.

- **Capítulo 3.** Modelo matemático del Quadcopter. Se ha realizado un modelado con ecuaciones matemáticas sobre las fuerzas y momentos a los que está sometido un Quadcopter en configuración X. Se presenta el modelado paso a paso mostrando imágenes exploratorias al mismo tiempo que definiciones para un fácil entendimiento.

- **Capítulo 4.** Algoritmo propuesto. Este capítulo es donde se muestra la idea de controlador para solventar el problema del aterrizaje autónomo. En primer lugar se muestra una introducción y posteriormente se presenta un diagrama de cajas. Y el pseudocódigo para finalizar esta parte.

- **Capítulo 5.** Experimentos. Cuando el lector esté en este capítulo verá en primer lugar la plataforma en la cual se ha programado el controlador, el idioma de programación, los entornos utilizados (ROS), la controladora PX4 y el ordenador de abordo Odroid. Posteriormente se explican los experimentos que se han realizado en condiciones de exterior e interior.

- **Capítulo 6.** Resultados. En este apartado se muestran los resultados de los experimentos realizados explicados en el capítulo 5, y la conclusión que sacamos de ellos.

- **Capítulo 7.** Trabajos Futuros. Para finalizar, en este capítulo se muestra al lector que se puede desarrollar después de este trabajo para optimizar y mejorar el control del Autolanding en las mismas condiciones o añadiendo otras nuevas.

Capítulo 2

II. ESTADO DE LA TECNOLOGÍA

Uno de los campos en los que más se ha trabajado dentro de los Quadcopter es en el control del aterrizaje autónomo. Hoy en día, los Quadcopter carecen de una gran radio de acción, han de volver a la plataforma para recargar baterías, o en el caso que se queden sin estas en plena misión, que puedan aterrizar de forma autónoma sobre algún lugar sin dañar la estructura del Quadcopter en sí. En esta parte analizaremos algunas de las tecnologías de sistemas de control de aterrizaje autónomo que se han desarrollado hasta el momento.

Los controladores de un Quadcopter pueden ser muy diferentes en estructura, pero todos ellos buscan que el Drone mantenga un vuelo estable, y pueda llegar a aterrizar de forma segura en el punto deseado. Cada sistema obtiene los datos del entorno mediante diferentes medios, como pueden ser; cámaras que procesan algoritmos para conocer la posición de la plataforma de aterrizaje, sensores internos inerciales [10], GPS, DGPS [8,4]… Todos ellos son válidos, aunque en nuestro caso utilizaremos cámara para la detección del 'April Tag' [6], Ultrasonidos para conocer la altura del Drone respecto al threshold, y sensores inerciales para saber el empuje que tiene que hacer cada uno de los motores para realizar un control adecuado de pitch, Roll y Yaw.

Controladores LQR y SMC para el aterrizaje autónomo

Este sistema [2] de control de implementa en un Drone del tipo Quadcopter que despega desde una plataforma con capacidad de movimiento independiente (UGV), con el objetivo de aportar al UAV una mayor libertad de movimiento y una mayor capacidad de autonomía, debido a que cuando este termina su misión, o se queda sin batería puede aterrizar sobre esta plataforma la cual le ha seguido. Sin necesidad de volver a una supuesta base de despegue alejada.

La tecnología que se implementa consiste en dos controladores del tipo LQR (Linear Quadraic Regulator) y dos SMC (Sliding Mode Controlller).

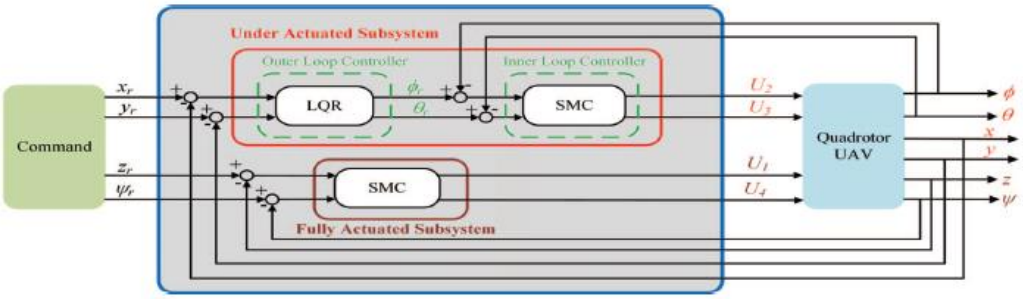

Figura II.1 Control con LQR y SMC [2].

Como se puede observar en el esquema de control hay tres controladores, dos SMC y un LQR. El LQR está en el Bucle externo y se encarga de obtener las posiciones deseadas en X e Y, y las transforma en ángulos de Pitch y Roll, para que el Drone pueda desplazarse de forma suave. El Linear Quadratic Regulator se diferencia del PID en que es un método de control moderno y su funcionamiento se basa en espacio de estados, suele ser útil para sistemas de varias salidas. Por otro lado el SMC es el bucle interno y se encarga de converger los valores actuales de Pitch y Roll a los deseados obtenidos a partir del bucle externo.

Para el control del Yaw y el eje Z solo hace falta un Controlador SMC, el cual tiene la misma función que el anterior; calcular la diferencia entre el Yaw y Z actuales y realimentar el sistema con ese error, para una aproximación delicada. Como se muestra a continuación el SMC tiende a minimizar el valor final de la altitud y el Yaw.

$$\lim_{t\to\infty} \|e_z\| = \|z_r - z\| = 0$$

$$\lim_{t\to\infty} \|\varphi_z\| = \|\varphi_r - \varphi\| = 0$$

Ecuación II-1 Ecuaciones SMC Yaw y altitud [2].

En el caso de este proyecto al aplicar un PID, para cada una de las entradas, se podrá conseguir uniformidad en nuestro sistema al no combinar diferentes tipos de controladores.

Control Switcher, dos fases de descenso y aterrizaje

Este control de aterrizaje está basado en una estrategia de cambio de control de largo recorrido a uno de corto recorrido, para optimizar y suavizar la aproximación del UAV [6]. Cuando desciende a partir de un

valor pre-programado de altura, 'Z Threshold', cambia de un controlador de Largo recorrido a uno de corto recorrido.

Sistema de Control de Largo recorrido [6]: La unidad de procesamiento gráfico (GPU) genera los comandos de la velocidad y posición. Un controlador Integral Proporcional se encarga de las coordenadas de la posición. Para suavizar la sobreoscilación que este genera, se le suma una señal proveniente de un controlador derivativo de la velocidad.

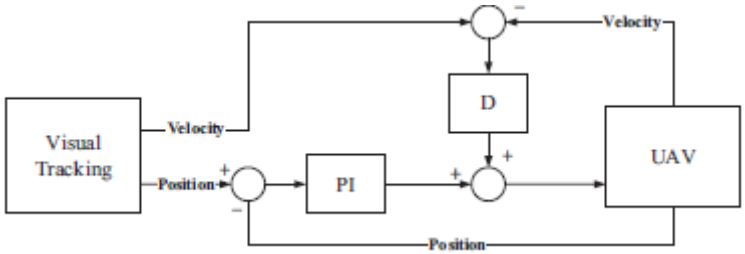

Figura II.2 Control de larga distancia [6].

En el momento en el que la distancia entre el UAV y la plataforma es inferior a la 'Z threshold', el control cambia a uno de corto radio, el cual es más preciso, y ayudará a orientar el UAV Quadcopter en una posición determinada en la plataforma [6].

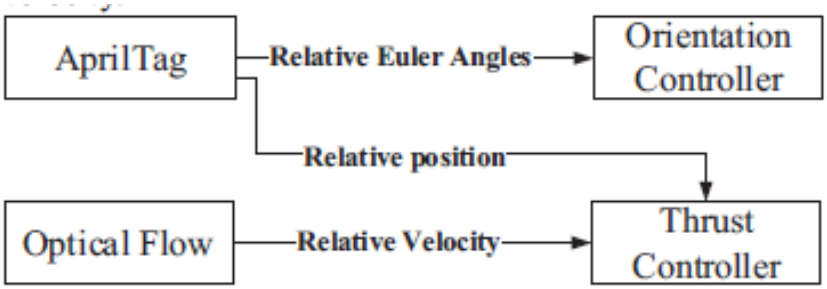

Figura II.3 Control de Corta distancia [6].

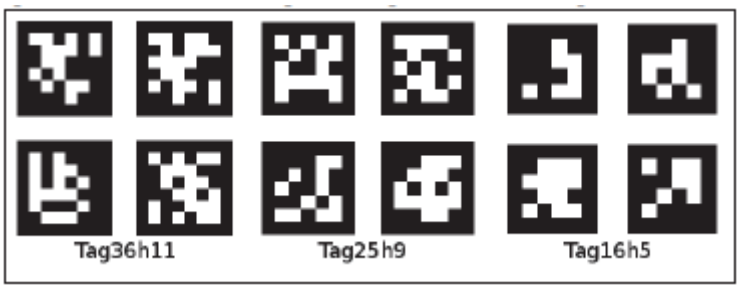

Figura II.4 Arpil Tag ejemplos [6].

El April Tag es el patrón de aterrizaje compuesto por pixeles negros y blancos. A partir de la visión computarizada se puede detectar a partir

del April Tag, a que ángulos queremos orientar el UAV, para ello manda potencia para que se modifique el Yaw, a su vez, el April Tag nos manda información de su posición, para que se puedan modificar el empuje de cada uno de los motores en un controlador y pueda moverse el Quadcopter en los ejes X, Z e Y. La velocidad relativa del UAV con respecto a la plataforma (porque ésta puede ser móvil), se calcula a partir de llamado 'Optical Flow', el cual es un patrón del movimiento aparente de los objetos, superficies y bordes entre un observador 'cámara' y un entorno [7].

Esta tecnología puede ser útil para aterrizar de forma muy precisa sobre un 'setpoint' predeterminado, pero también puede llegar a tener problemas derivados del cambio instantáneo de un controlador a otro, ya que puede ser un punto débil del sistema, por ello para un UAV optimo y barato, lo más aconsejable es apostar por un único controlador para el aterrizaje.

Control Borroso (Fuzzy) para el aterrizaje

Este tipo de controlador [4] es muy innovador ya que se basa en el Control Inteligente, la tecnología borrosa pretende imitar las decisiones humanas las cuales son información continua, y también contempla la mejora de métodos basados en la lógica binaria.

La visión computarizada se ha ido introduciendo en el control de sistemas de vehículos para poder utilizar la información que puede aportar, una de estas aplicaciones es el aterrizaje autónomo.

El control borroso (Fuzzy), analiza la información visual que se recibe de la cámara para generar un camino de descenso. La posición del UAV en este caso se recibe mediante un GPS, e IMU. Los datos se mandan a un control Fuzzy, que hace un control de altitud y ulteriormente se hace posible el aterrizaje autónomo.

El controlador Fuzzy tiene dos entradas y una salida [4]:

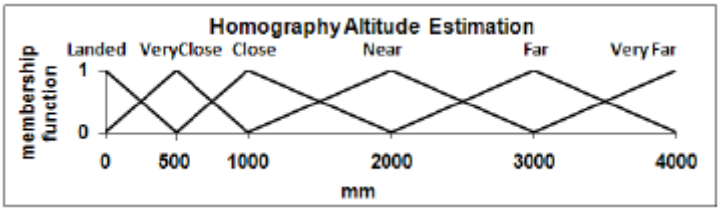

Figura II.5 Entrada 1 estimación de la altitud [4].

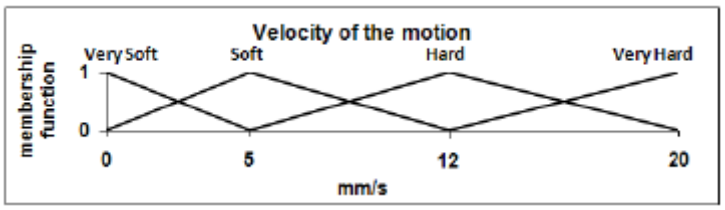

Figura II.6 *Entrada 2. Variación de la altitud con respecto a los últimos dos frames capturados [4].*

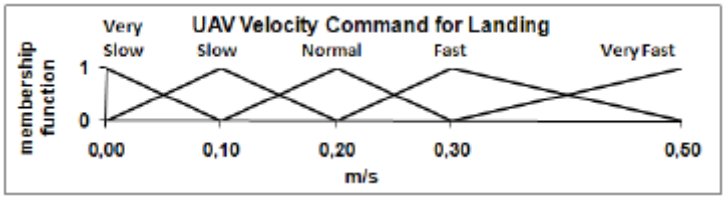

Figura II.7 Salida del controlador Fuzzy [4].

La salida del control es el comando de la velocidad en metros por segundo que son ejecutados por el UAV a medida que se aproxima a la plataforma de aterrizaje.

Además este controlador Fuzzy tiene a la salida un PID, que funciona como controlador a nivel bajo para asegurar la estabilidad, los errores calculados por el PID, retroalimentan al sistema y vuelven a dar los comandos necesarios al control Fuzzy para una nueva velocidad de descenso.

A continuación se expone el esquema completo de control de esta tecnología borrosa más PID:

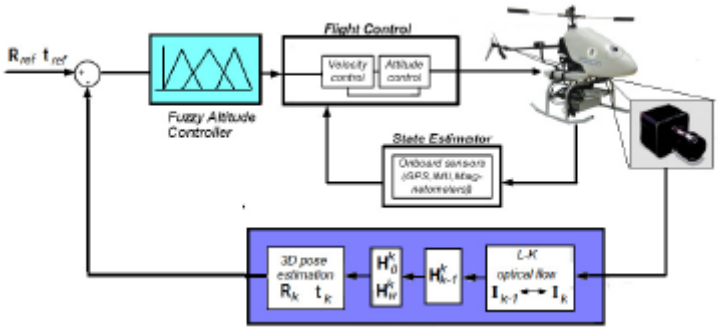

Figura II.8 Control borroso más PID para el control del aterrizaje [4].

El control borroso es muy útil para controlar la no linealidad que tiene el entorno de operación del UAV. En un PID puro a veces los coeficientes calculados por el método manual de Ziegler-Nichols no están propiamente calculados y no se hace un control del todo preciso [12].

Por eso se requiere un Fuzzy capaz de modificar los parámetros de las ganancias del PID automáticamente como se muestra en la siguiente figura:

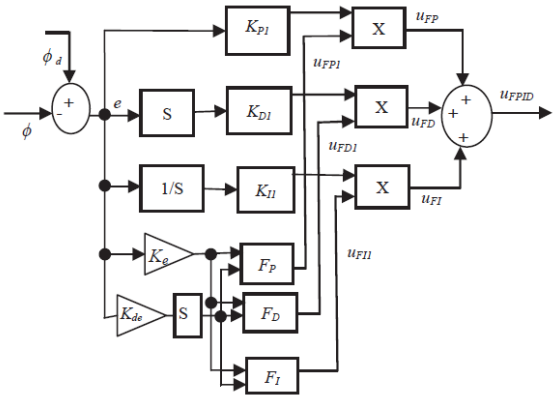

Figura II.9 Funcionamiento básico de un control Borroso [12].

En la figura anterior (únicamente para el control del ángulo Roll) se ve que los bloques F_p, F_d, y Fi son el control Fuzzy. K_{p1},K_{i1}, K_{d1},K_e,K_{de} son constantes que se utilizan como factor de escala, que posteriormente se multiplican por K_{p1},K_{i1}, y K_{d1} para crear las señales de control final $u_{fp}(k)$, $u_{fi}(k)$, y $u_{fd}(k)$, que son usadas para configurar los parámetros del PID finales. La misma estructura es utilizada para el Pitch y Yaw [12].

Control PID en eje vertical

Como en la mayoría de los sistemas de aterrizaje autónomo, para encontrar la posición en la que se encuentra el Drone se utiliza visión computarizada, Se usa un controlador PID en X e Y que recibe los comandos de X final e Y final, de donde está la plataforma de aterrizaje y se la información se manda a los movimientos pitch y Roll para alcanzarla, cuando el Quadcopter está justamente encima del patrón de aterrizaje, comienza a descender paso a paso en el eje de la altura, Sí la cámara detecta que el patrón se ha desviado , vuelve a mandar potencia a los motores hasta recuperar el patrón en la visión de la cámara.

Básicamente este control consiste en situarse sobre la plataforma de aterrizaje e ir actualizando el valor de Z, 'step by step' hasta que está a una altura de seguridad de 5 cm (hasta 15 cm) [5] de la plataforma en la que puede apagar los motores.

A continuación se puede observar el esquema utilizado por [5] en esta tecnología:

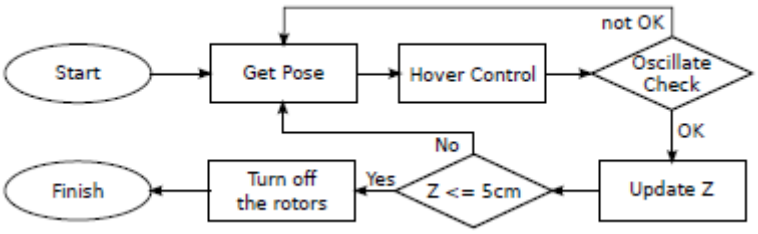

Figura II.10 Esquema de PID en eje vertical [5].

Tianqu Zhao y Hong Jiang [22] también incorporan este sistema de control a un Quadcopter Ar.Drone 2.0. Ya que es uno de los más fiables debido a que es muy simple y eficaz. En este caso el Drone posee dos cámaras, una de visión horizontal (que no se utiliza en el aterrizaje), otra vertical que mira hacia abajo, tres sensores giroscópicos y tres acelerómetros.

Como en otros muchos casos, los 'frames' de la imagen, son captados y procesados por un paquete de reconocimiento de marcadores de ROS, y calculan la posición y orientación del Drone y la trayectoria de aproximación. Ulteriormente se usa un controlador PID como en [5] para controlar la velocidad y estabilidad del Quadcopter en el recorrido de aproximación, por lo tanto, se controla el comportamiento dinámico del sistema.

$$U_t = K_P\left(X_{(0,0)} - X_{t-1}\right) + K_D\left(\dot{X}_{(0,0)} - \dot{X}_{t-1}\right) + K_I \int_0^t (X_{(0.0)} - X_{t-1})dt$$

Ecuación II-2 Función de un PID [22].

U_t es la salida del controlador, X_{t-1} es la posición del Drone en un instante anterior, y X_{0-0} es el setpoint que está en el centro de la imagen-objetivo procesada por la estación de procesado. Antes de que el Drone descienda de altura se necesita aplicar el controlador PID para los ejes X y Y para controlar la velocidad lineal en X y velocidad lineal en Y. Así se asegura que la aproximación es exitosa y no hay desvíos en estos ejes. Posteriormente cuando este procesa ha concluido, se aplica el PID en el eje Z y se va anulando el error de la entrada con respecto a la salida del sistema de control. Este proceso se repite hasta que se alcanza una altura predeterminada de 'threshold' y entonces se da por satisfactorio el aterrizaje.

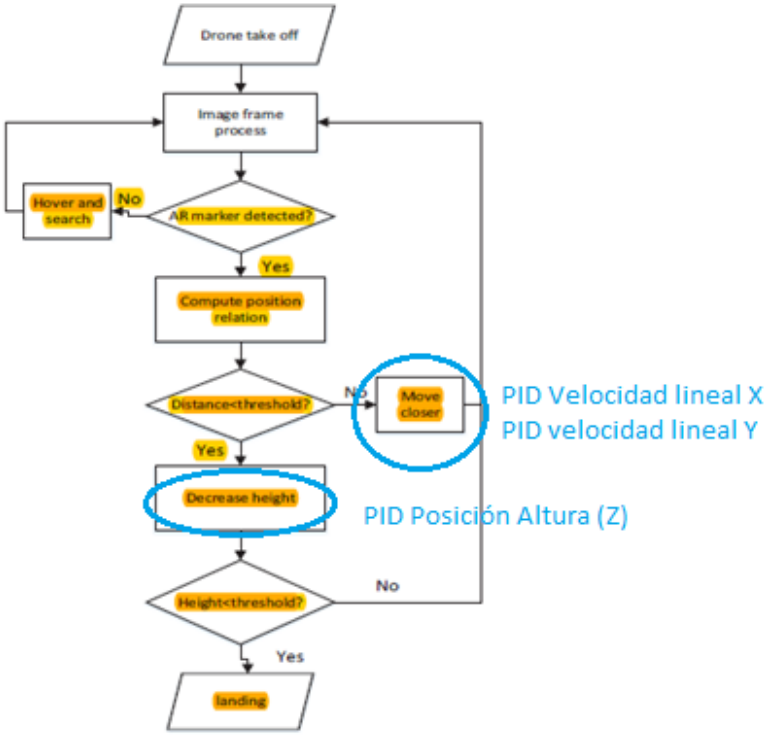

Figura II.11 Flow Chart Extraído de [22].

Hacer un controlador únicamente en Z para la fase de descenso, tiene sus ventajas, como es el aterrizaje vertical (VL), al igual que es muy simple y por lo tanto menos expuesto a fallos, pero también tiene sus inconvenientes como es la carencia de hacer un aterrizaje y una aproximación suave y progresiva desde una distancia en X, e Y desplazadas de la plataforma de aterrizaje, mediante este último tipo de aterrizaje se gasta menos batería ya que los motores requieren de menos potencia para hacer un 'Hovering' 'Step by Step', otro de los problemas que pueden surgir a partir del último diseño [22], es a partir del procesado de imágenes por cámara y posterior envío de los datos por vía Wi-Fi. Debido a que pueden enviar señales con ruido al igual que en [3], por lo tanto habría que mejorarlo colocando el procesado de imágenes a bordo del Drone en el caso que se necesite mucha precisión.

Control Backstepping para el aterrizaje

La idea de este tipo de control [8], es dividir la aproximación o descenso en cuatro fases distintas, cada fase cambia a la siguiente mediante un valor determinado que apartan los instrumentos y sensores de abordo como pueden ser el GPS diferencial, de gran precisión o el IMU. Para conocer la altitud se utilizan sensores tipo LIDAR (Laser Imaging

Detection and Ranging). La frontera entre cada una de las fases del aterrizaje se denomina de la siguiente manera [8]:

- Fase de aceleración constante (t_0-t_1)
- Fase de velocidad constante (t_1-t_2)
- Fase de deceleración constante (t_2-t_3)
- Fase de velocidad constante (t_3-t_{touch}), el comando de velocidad constante justo antes de posarse en la plataforma de aterrizaje.

Estas constantes se determinan a partir de unas ecuaciones que se denominan 'Path parameter computation' [8].

Y por último la controladora del UAV, resuelve las siguientes ecuaciones cinemáticas para cada una de las fases de aterrizaje:

$$\boldsymbol{si\ t\ \in [t_0; t_1]}$$

$$\ddot{z}(t) = a_{máxima}$$
$$\dot{z}(t) = a_{máxima} \cdot t$$
$$z(t) = \frac{a_{maxima}}{2} \cdot t^2 + z_0$$

$$\boldsymbol{si\ t\ \in [t_1; t_2]}$$

$$\ddot{z}(t) = 0$$
$$\dot{z}(t) = v_{máxima}$$
$$z(t) = v_{máxima} \cdot (t - t_1) + z_1$$

$$\boldsymbol{si\ t\ \in [t_2; t_3]}$$

$$\ddot{z}(t) = a_{mínima}$$
$$\dot{z}(t) = a_{min} \cdot (t - t_2) + v_{máxima}$$
$$z(t) = \left(\frac{-a_{mínima} \cdot (t - t_2)}{2} + v_{máxima}\right)(t - t_2) + z_2$$

$$\boldsymbol{si\ t\ \in [t_1; t_2]}$$

$$\ddot{z}(t) = 0$$
$$\dot{z}(t) = v_{final}$$
$$z(t) = v_{final} \cdot (t - t_3) + z_3$$

Ecuación II-3 Ecuaciones del BackStepping.

Este tipo de control consigue optimizar la trayectoria de aproximación con unos resultados muy precisos, El problema que ocurre es que el sensor Lidar Lite no es capaz de medir distancias inferiores a 100 mm, aproximadamente lo mismo que los sensores de ultrasonidos, siendo estos mucho más económicos. Por ello en este proyecto se utilizan este tipo de sensores.

Control PID con variación de ganancias Automática

Esta tecnología [3] se utiliza en algo que está muy requerido hoy en día, el aterrizaje autónomo de un UAV sobre un vehículo marino denominado 'Autonomous Kayak'. Es muy útil para misiones de rescate en alta mar, donde un Drone está muy lejos de la base más cercana, o para tareas de observación medioambientales. Para esta tecnología ha sido desarrollado una GCS (Ground Control System), la cual procesa las imágenes del video que el Drone aporta y manda los comandos para un aterrizaje seguro, toda esta transmisión de datos se hace vía Wi-Fi.

La plataforma de aterrizaje contiene patrones t colores distintivos para que se procese más fácilmente la imagen. La detección del patrón se basa en el procesamiento del color de la plataforma de aterrizaje.

El control es una versión modificada del PID, este tiene dos entradas Pitch y Roll.

$$roll = K_{P1} \cdot X_{error} + K_{D1} \cdot \dot{X}_{error} + K_{I1} \cdot \int_0^t X_{error}$$

$$pitch = K_{P1} \cdot Y_{error} + K_{D1} \cdot \dot{Y}_{error} + K_{I1} \cdot \int_0^t Y_{error}$$

Ecuación II-4 Ecuaciones para PID [3].

Uno de los inconvenientes de este tipo de vehículos es que están expuestos a las condiciones meteorológicas; el viento puede hacer variar en gran medida el comportamiento del controlador PID, por ello se ha desarrollado la tecnología adaptativa para las ganancias del controlador para contrarrestar el offset que el viento genera.

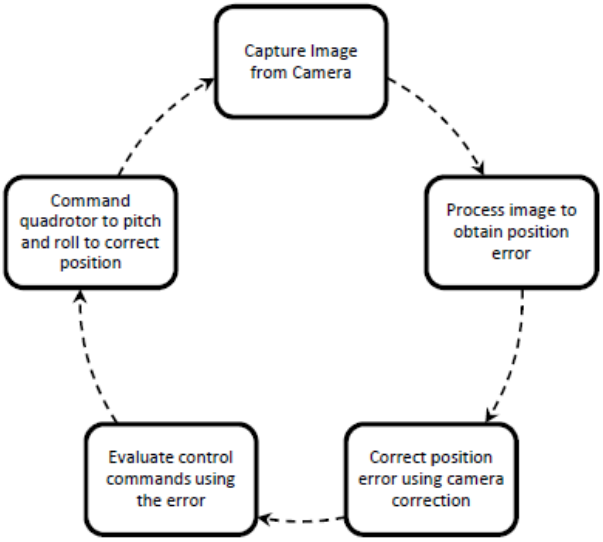

Figura II.12 Proceso de corrección que utiliza el sistema para el aterrizaje sobre la plataforma [3].

El problema que puede surgir a partir de esta tecnología es que la comunicación vía Wi-Fi, puede transmitir un alto nivel de ruido, por lo tanto el GCS, debería de instalarse dentro del UAV, aun sabiendo que se pudiese aumentar el peso, el aterrizaje sería más preciso.

Dead Reckoning con un Control PD

Dead Reckoning es un método muy habitual hoy en día para conocer la posición relativa de un UAV, consiste en obtener los datos de velocidad y rumbo actual a partir de sensores internos en el UAV, e introducirlos en fórmulas trigonométricas que se procesan. Es una alternativa a los GPS o DGPS, en caso que estos no reciban señal porque el UAV esté en sitios interiores [10].

Se busca utilizar estos datos recibidos con tecnología Dead Reckoning para controlar un aterrizaje autónomo de un Drone [1]. Para detectar la zona de aterrizaje se utiliza una cámara de abordo y la imagen es procesada mediante la GCS.

Esta tecnología busca [1] controlar la posición relativa del drone mediante el Dead Reckoning y controla los valores de x, y, z, y ángulo a través de 4 controladores Proporcional-Derivativos (PD), el esquema se muestra a continuación:

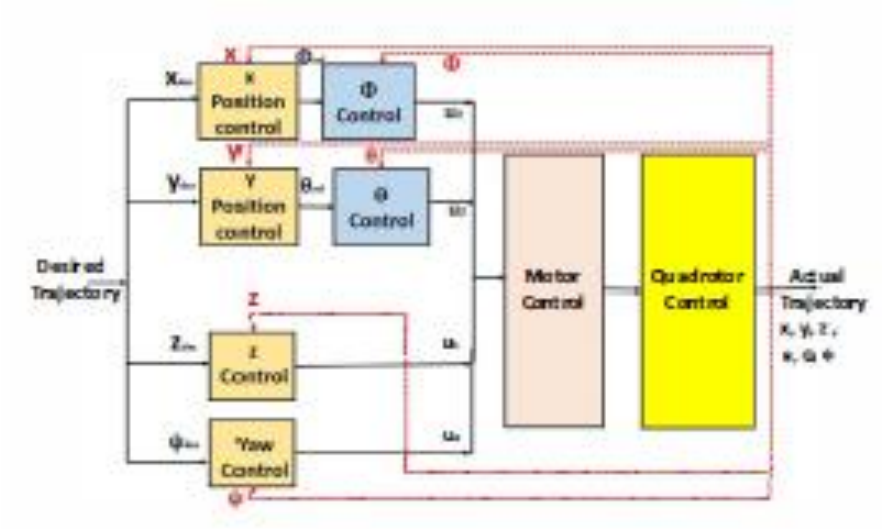

Figura II.13 Esquema de control, cuatro Controladores PD (x, y, z, ángulo) [1].

El sistema se retroalimenta para ir eliminando el error de distancia y orientación a medida que se aproxima a la plataforma.

Uno de los motivos por los cuales utilizar PD en vez de PID, es porque los subsistemas de X, Y, Z, y la orientación tienen dos ecuaciones de segundo orden, los controladores PID requerirían un orden de perfil movimiento más alto (una entrada al sistema más suave), como mínimo de orden 3, más que un controlador PD que le vale con perfiles de movimiento de segundo orden [11].Además un PID al requerir ecuaciones de movimiento de un orden mayor, requieren mayores aceleraciones y por lo tanto un mayor consumo de baterías. Pero en el PID, tenemos el término integral que puede ayudar a acelerar el proceso de aproximación al 'setpoint', con una ganancia Ki adecuada, que vamos a necesitar en este proyecto.

Control óptimo PID Con algoritmo basado en DE.

La técnica del 'Differential Evaluation' (DE), puede generar una solución a partir de una gran calidad de cálculos, en un menor tiempo de procesado, uno de los atractivos del sistema DE incluye que no es necesario aportar ninguna información adicional del gradiente [12]. Lo único que hay que hacer para que se compute bien el proceso es asignar un NP (Numero de partículas), un factor D (Numero de parámetros de la función a optimizar). Y mediante unas fórmulas evolutivas [12-C], va mutando el problema hacia un determinado objetivo.

El DE se utiliza para optimizar los parámetros del controlador PID, para regular la estabilidad del movimiento del UAV (Roll, Pitch, Yaw) [12].

Básicamente el DE se basa en tres diferentes fases: Mutación, cruzamiento y sección [12]. El controlador que proponen para esta tecnología se puede observar en la siguiente figura:

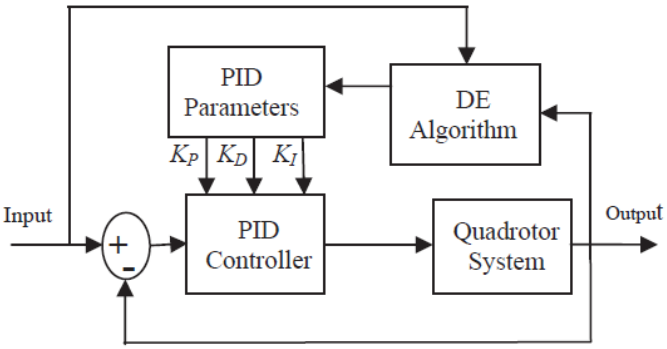

Figura II.14 Controlador DE-PID [12].

Como se observa en la Figura, la entrada al bloque DE, son los vectores Roll, Pitch y Yaw de referencia, y el vector actual de los mismos. El bloque optimiza los parámetros de Kp, Ki y Kd para cada ángulo [12]. El trabajo que se realizará en este únicamente tendrá un bloque de PID, para cada uno de los valores pitch roll and yaw, dejando algoritmos evolutivos para trabajos futuros.

Control Adaptativo usando Backstepping y Superficie de Control Dinámica

Jawhar Ghommam y Maarouf Saad [23] han desarrollado un control de aterrizaje que se basa en aterrizar un Quadrotor en una plataforma móvil combinando varias fases de aproximación, procesado de datos y aterrizaje para lograr el objetivo final.

En primer lugar el Quadrotor, se encuentra a una distancia considerable del punto de aterrizaje, por lo tanto no puede detectarla aun, en este punto el GPS es utilizado como herramienta de posicionamiento ya que no se requiere mucha precisión, entonces calcula un punto de destino en el límite de una esfera virtual de radio predeterminado y con centro la plataforma móvil de aterrizaje con el fin de poder detectar. En el momento que este punto es interceptado, la medición de la posición hace un cambio y pasa a medir la posición relativa del patrón de aterrizaje con la cámara ya que el GPS es menos preciso, a continuación cambia de colocarse en el punto límite de la esfera virtual a un punto vertical del patrón para proceder al aterrizaje en aproximación vertical después de haber recibido una señal de autorización. A continuación se muestra un esquema de control utilizado en [23].

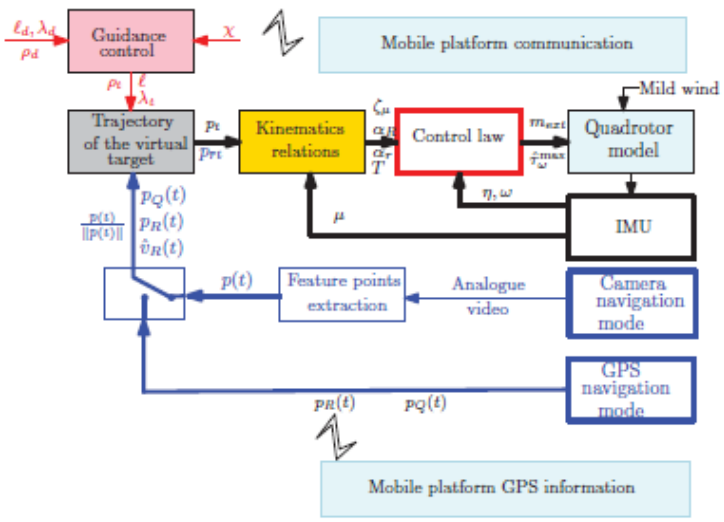

Figura II.15 Esquema de control Backstepping + DSC [23].

Como se puede observar en la figura anterior, existe una posibilidad de cambio de medida de posición relativa del Drone, dependiendo si está dentro o fuera de la esfera virtual que rodea la plataforma de aterrizaje móvil.

Resumen del estado del Arte.

En este capítulo se han mostrado los estudios, y lecturas que se han realizado a partir de papers de investigación sobre los sistemas de aterrizaje autónomo de Quadcopters. Principalmente nos hemos centrado en buscar sistemas de control como por ejemplo controladores PID, PD, control óptimo, controladores LQR y control Fuzzy.

La función de esta investigación sobre la tecnología que se emplea, constituye una parte importante del proyecto ya que es la base desde la cual desarrollamos nuestro estudio.

En cada tecnología aparece una o varias referencias, sobre las personas que realizaron la investigación.

Capítulo 3

III. MODELO MATEMÁTICO DEL UAV

Desarrollo de las ecuaciones

El Quadcopter es una aeronave que tiene cuatro motores, estos están situados a una distancia d del centro de gravedad (CDG). A continuación se mostrará paso a paso, el estudio del modelo dinámico de un Quadcopter en configuración X.

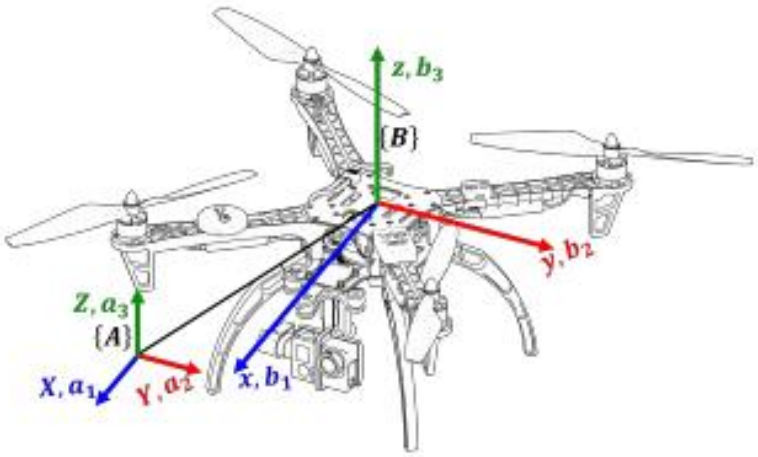

Figura III.1 Sistemas de referencia del Quadcopter en configuración X. Global y local.

Como en todo solido rígido en movimiento, en un Quadcopter se definen dos sistemas de referencia; el global, el cual se sitúa fuera de la estructura del mismo, y el local, que es el que va siguiendo el movimiento del UAV, normalmente se suele situar en su CDG [19].

$$\boldsymbol{X} = \begin{pmatrix} \boldsymbol{x} \\ \boldsymbol{y} \\ \boldsymbol{z} \end{pmatrix} \quad (1)$$

$$\boldsymbol{X_b} = \begin{pmatrix} \boldsymbol{x_b} \\ \boldsymbol{y_b} \\ \boldsymbol{z_b} \end{pmatrix} \quad (2)$$

El UAV tiene seis grados de libertad (6DOF), están designados por los movimientos de traslación {x, y, z} y por los movimientos de rotación

{φ, θ, ψ}, roll pitch y Yaw respectivamente. En la figura siguiente se muestran esquemáticamente:

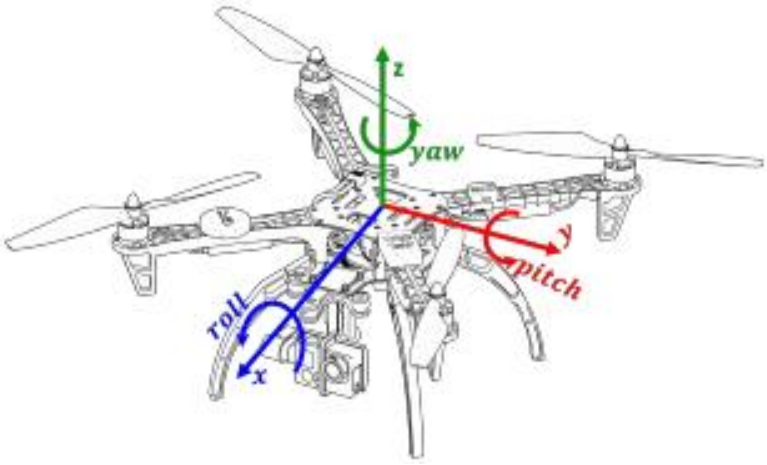

Figura III.2 Pitch, Yaw y Roll en configuración X.

Para poder definir correctamente la matriz de inercia de una UAV se tienen que desarrollar anteriormente el concepto de empuje y el de momento.

Empuje es la fuerza que ejercen los motores del Quadcopter para que este pueda elevarse y volar. El valor de empuje de un motor se obtiene de la siguiente ecuación [13]:

$$\boldsymbol{Empuje = c_T \cdot A_R \cdot \rho \cdot r^2 \cdot \bar{\omega}^2} \quad (3)$$

Donde c_T es el coeficiente de empuje, A_R, es el área de la hélice cuando está en rotación, ρ, es la densidad del fluido del entorno, en este caso aire, $\bar{\omega}^2$, es la velocidad angular de los motores, y r^2, es el ratio del rotor.

La ecuación (3) se puede simplificar y contraer y quedaría de la siguiente manera:

$$\boldsymbol{Empuje = C_T \cdot \bar{\omega}^2} \quad (4)$$

Con C_T , siendo el coeficiente de empuje agrupado (agrupando todas las constantes de la ecuación (3)).

El mismo proceso se podría hacer para la fórmula del momento que cada rotor genera:

$$\boldsymbol{Momento = Q_T \cdot \bar{\omega}^2} \quad (5)$$

Donde Q_T, es el coeficiente agrupado del Momento.

Como se puede observar en las ecuaciones (4) y (5):

$$\boldsymbol{Empuje \propto \bar{\omega}^2} \quad (6)$$

$$\boldsymbol{Momento \propto \bar{\omega}^2} \quad (7)$$

Son directamente proporcionales a las revoluciones por minuto (RPM) del motor, por lo tanto el sistema no puede ser directamente controlado mediante los comandos que se utilizan. Para arreglar esto, se hace una aproximación mediante regresión [13] como se muestra a continuación:

$$\boldsymbol{\bar{\omega}_{Regresión} = (Empuje\ \%) \cdot C_R + b} \quad (8)$$

Donde $\bar{\omega}_{Regresión}$, es la velocidad del rotor, ($Empuje\ \%$), es la entrada del comando del empuje en tanto por ciento, C_R, es el coeficiente de conversión, y b, es la ordenada en el origen de la regresión.

En nuestro caso tenemos cuatro rotores simétricos en X, por lo tanto, se desarrollan las ecuaciones para el modelo dinámico:

$$\boldsymbol{\sum_{i=1}^{N} Empuje = C_T \cdot \sum_{i=1}^{N} \bar{\omega} i^2} \quad (9)$$

La ecuación (9) es el empuje total del Quadcopter, en nuestro caso $N = 4$.

Figura III.3 Fuerza Empuje con la numeración de cada uno de los rotores [21].

Posteriormente se tienen que obtener los momentos totales con respecto al pitch θ, roll φ y Yaw ψ [13]. Por definición la ecuación para calcular el momento de una fuerza es:

$$\sum \overline{\boldsymbol{Momentos}} = \sum(\bar{\boldsymbol{r}} \times \bar{\boldsymbol{F}}) \quad (10)$$

Donde $\bar{r}$, es la distancia a donde se aplica la fuerza y F, la fuerza en sí. Por lo tanto para calcular los momentos que generan el roll (μ_φ), pitch (μ_θ) y Yaw (μ_ψ) se ha de multiplicar la fuerza que genera el empuje por la distancia a cada uno de los rotores para el movimiento de Pitch y Roll. En cambio para ver el momento del Yaw, nos apoyamos en la ecuación (5).

$$\boldsymbol{\mu_\varphi = C_T \cdot \sin \alpha_i \cdot d \cdot \omega_i^2} \quad (11)$$

$$\boldsymbol{\mu_\theta = -C_T \cdot \cos \alpha_i \cdot d \cdot \omega_i^2} \quad (12)$$

$$\boldsymbol{\mu_\psi = Q_T \cdot \sigma_i \cdot \omega_i^2} \quad (13)$$

Se denomina α_i, al ángulo que hace el brazo del Quadcopter con respecto al eje positivo del sistema de referencia local r_b. Y σ_i , es el sentido de giro de los rotores.

Se puede construir la siguiente matriz, la cual representa un modelo dinámico de un Quadcopter en configuración X basándonos en las ecuaciones anteriores (9), (11), (12) y (13):

$$\begin{pmatrix} \sum_{i=1}^{4} \boldsymbol{Empuje} \\ \boldsymbol{\mu_{\varphi}} \\ \boldsymbol{\mu_{\theta}} \\ \boldsymbol{\mu_{\psi}} \end{pmatrix} = \begin{pmatrix} \boldsymbol{C_T} & \boldsymbol{C_T} & \boldsymbol{C_T} & \boldsymbol{C_T} \\ \boldsymbol{C_T \sin \alpha_i \, d} & \boldsymbol{C_T \sin \alpha_i \, d} & \boldsymbol{C_T \sin \alpha_i \, d} & \boldsymbol{C_T \sin \alpha_i \, d} \\ \boldsymbol{-C_T \cos \alpha_i \, d} & \boldsymbol{-C_T \cos \alpha_i \, d} & \boldsymbol{-C_T \cos \alpha_i \, d} & \boldsymbol{-C_T \cos \alpha_i \, d} \\ \boldsymbol{Q_T \sigma_i} & \boldsymbol{Q_T \sigma_i} & \boldsymbol{Q_T \sigma_i} & \boldsymbol{Q_T \sigma_i} \end{pmatrix} \begin{pmatrix} \overline{\boldsymbol{\omega_1^2}} \\ \overline{\boldsymbol{\omega_2^2}} \\ \overline{\boldsymbol{\omega_3^2}} \\ \overline{\boldsymbol{\omega_4^2}} \end{pmatrix} \quad (14)$$

Agrupando términos y desarrollando la ecuación anterior con los términos de α_i=45º, $\sigma_i = 1$ para los motores 1, y 3 $\sigma_i = -1$, para los números 2 y 4. Y definiendo d_x, como la distancia ya descompuesta en los ejes de referencia del Quadcopter.

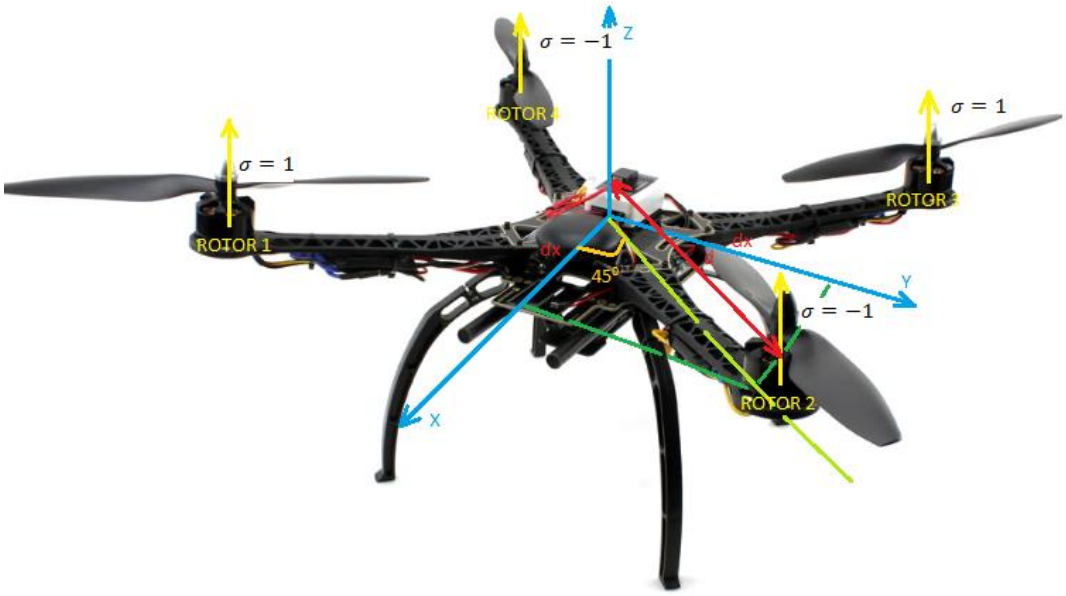

Figura III.4 Esquema gráfico de los elementos del UAV en configuración X [21].

Nos quedaría una ecuación como la siguiente:

$$\begin{pmatrix} \sum_{i=1}^{4} \boldsymbol{Empuje} \\ \boldsymbol{\mu_{\varphi}} \\ \boldsymbol{\mu_{\theta}} \\ \boldsymbol{\mu_{\psi}} \end{pmatrix} = \begin{pmatrix} \boldsymbol{C_T} & \boldsymbol{C_T} & \boldsymbol{C_T} & \boldsymbol{C_T} \\ \boldsymbol{C_T d_x} & \boldsymbol{-C_T d_x} & \boldsymbol{C_T d_x} & \boldsymbol{-C_T d_x} \\ \boldsymbol{-C_T d_x} & \boldsymbol{-C_T d_x} & \boldsymbol{C_T d_x} & \boldsymbol{C_T d_x} \\ \boldsymbol{Q_T} & \boldsymbol{-Q_T} & \boldsymbol{Q_T} & \boldsymbol{-Q_T} \end{pmatrix} \begin{pmatrix} \overline{\boldsymbol{\omega_1^2}} \\ \overline{\boldsymbol{\omega_2^2}} \\ \overline{\boldsymbol{\omega_3^2}} \\ \overline{\boldsymbol{\omega_4^2}} \end{pmatrix} \quad (15)$$

La ecuación (15) es una matriz que define el modelo matemático de un Quadcopter en configuración X.

Aparición de un efecto giroscópico [20]: es un fenómeno que aparece cuando se somete un cuerpo con elementos en movimiento de rotación a un momento de fuerza que tiende a cambiar la orientación del eje de rotación. Le afecta a los momentos que se generan por el pitch y Yaw [13]:

$$\boldsymbol{\mu_{\varphi_{Giro}} = J_m Q\left(\pi/_{30}\right)(\omega_1 - \omega_2 + \omega_3 - \omega_4)} \quad (16)$$

$$\boldsymbol{\mu_{\theta_{Giro}} = J_m P\left(\pi/_{30}\right)(-\omega_1 + \omega_2 - \omega_3 + \omega_4)} \quad (17)$$

En las ecuaciones (16) y (17) J_m, es el tensor de inercia, Q y P son los valores angulares del Pitch y Roll, y $\left(\pi/_{30}\right)$, es el valor que convierte el valor de revoluciones a radianes.

Finalmente nos queda la ecuación (19) que define el modelo dinámico del Quadcopter en configuración X, teniendo en cuenta el par giroscópico que se genera.

$$\boldsymbol{M_{total} = \begin{pmatrix} \mu_{\varphi} + \mu_{\varphi_{Giro}} \\ \mu_{\theta} + \mu_{\theta_{Giro}} \\ \mu_{\psi} \end{pmatrix}} \quad (18)$$

$$\boldsymbol{M_{total} = \begin{pmatrix} C_T\overline{\omega_1^2}d_x - C_T\overline{\omega_2^2}d_x + C_T\overline{\omega_3^2}d_x - C_T\overline{\omega_4^2}d_x + J_m Q\left(\pi/_{30}\right)(\omega_1 - \omega_2 + \omega_3 - \omega_4) \\ -C_T\overline{\omega_1^2}d_x - C_T\overline{\omega_2^2}d_x + C_T\overline{\omega_3^2}d_x + C_T\overline{\omega_4^2}d_x + J_m P\left(\pi/_{30}\right)(-\omega_1 + \omega_2 - \omega_3 + \omega_4) \\ Q_T\overline{\omega_1^2} - Q_T\overline{\omega_2^2} + Q_T\overline{\omega_3^2} - Q_T\overline{\omega_4^2} \end{pmatrix}} \quad (19)$$

En (18) y (19) M_{total}, es el Momento total, que generan el movimiento de Pitch Roll y Yaw en la estructura del Quadcopter en un momento determinado.

Estabilizar el Quadcopter.

El control de la trayectoria de aterrizaje de un Quadcopter se realiza mediante un controlador PID, en la ecuación (21):

$$error = X_{final} - X_{actual} \quad (20)$$

$$Entrada\ al\ Sistema\ (Drone) = K_P(error) + K_D\frac{d(error(t))}{dt} + K_I\int_0^t error(t)\ dt \quad (21)$$

En este proyecto se realizarán tres controladores, uno para cada eje, por lo tanto existes tres entradas al sistema, estas entradas se denominarán comandos. Y debido a las características de la controladora PX4 PixHawk han de variar entre 1000 y 2000 siendo 1500 la posición de reposo. Las ecuaciones son la (22), (23) y (24), las cuales se muestran a continuación:

$$Comando(Throttle) = K_{Pz}(Zf - Z) + K_D\frac{d(Zf-Z)}{dt} + K_I\int_0^t Zf - Zdt \quad (22)$$

$$Comando\ Pitch\ (\theta) = K_P(Xf - X) + K_D\frac{d(Xf-X)}{dt} + K_I\int_0^t Xf - X\ dt \quad (23)$$

$$Comando\ Roll\ (\varphi) = K_P(error) + K_D\frac{d(error(t))}{dt} + K_I\int_0^t error(t)\ dt \quad (24)$$

Resumen modelo matemático del Drone

En primer lugar se menciona el tipo de Drone con el que vamos a trabajar: en configuración X. Posteriormente se presenta un esquema mostrando los sistemas de referencia del Drone; el global y el local siendo esenciales para la realización este proyecto.

A continuación con el fin de mostrar los momentos a los que se somete una estructura de Quadcopter en configuración X. El momento en cada uno de los ejes varía tanto con el peso y empuje (3), (4) y (5) como por el par giroscópico (16) y (17), que aparece cuando hay objetos en movimiento de rotación.

Finalmente se muestran las ecuaciones de un controlador PID que se han utilizado en los ejes 'X',' Y' y 'Z', para calcular los comandos Pitch, Roll y Throttle respectivamente dentro de un rango de 1000-2000.

Capítulo 4

IV. ALGORITMO PROPUESTO

En este apartado se muestra la lógica del algoritmo implementado. Este algoritmo se ha creado en lenguaje de programación C++ y ROS. Para un mejor entendimiento se presenta un Flow Chart que muestra las fases de funcionamiento y posteriormente el pseudocódigo de las funciones del Ejecutable creado.

FlowChart

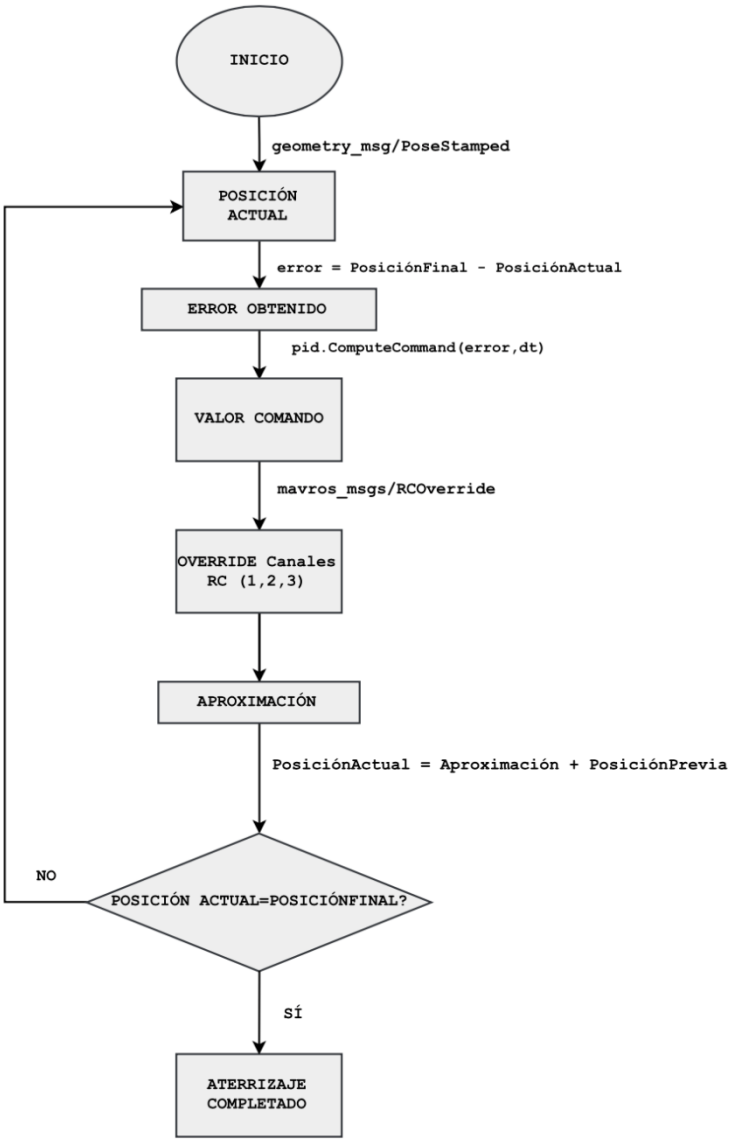

Figura IV.1 Flow-Chart del Funcionamiento del PID.

Pseudocódigo

Cada una de las funciones desarrolladas dentro del Controlador en código de C++ se muestran a continuación en forma de pseudocódigo para su mayor entendimiento:

Proceso: pidCalcularThrottle (param1)

Entrada: *Error en el eje Z.*

Salida: *Esfuerzo del Throttle que se ha de enviar a la controladora mediante MavLink.*

1. Comienzo

2. Tiempo Actual = Ahora

3. Esfuerzo = ComputarComando (error Z, Tiempo Actual – Tiempo anterior)

4. Si (esfuerzo > 500), devolver esfuerzo = 500

Si (esfuerzo < - 500), devolver esfuerzo = - 500

5. Tiempo anterior = Tiempo Actual

6. Esfuerzo = esfuerzo + Comando de equilibrio (1500)

7. Devolver esfuerzo

Proceso: pidCalcularRoll (param1)

Entrada: *Error en el eje Y.*

Salida: *Esfuerzo del Roll que se ha de enviar a la controladora mediante MavLink.*

1. Comienzo

2. Tiempo Actual = Ahora

3. Esfuerzo = ComputarComando (error Y, Tiempo Actual – Tiempo anterior)

4. Si (esfuerzo > 500), devolver esfuerzo = 500

Si (esfuerzo < - 500), devolver esfuerzo = - 500

5. Tiempo anterior = Tiempo Actual

6. Esfuerzo = esfuerzo + Comando de equilibrio (1500)

7. Devolver esfuerzo

Proceso: pidCalcularPitch (param1)

Entrada: *Error en el eje X.*

Salida: *Esfuerzo del Pitch que se ha de enviar a la controladora mediante MavLink.*

1. Comienzo

2. Tiempo Actual = Ahora

3. Esfuerzo = ComputarComando (error X, Tiempo Actual – Tiempo anterior)

4. Si (esfuerzo > 500), devolver esfuerzo = 500

Si (esfuerzo < - 500), devolver esfuerzo = - 500

5. Tiempo anterior = Tiempo Actual

6. Esfuerzo = esfuerzo + Comando de equilibrio (1500)

7. Devolver esfuerzo

Proceso: positionCallBack(param1)

Entrada: *Posición actual del Drone.*

1. Comienzo

2. Establecer cada uno de los canales del RC = 0

3. Error de posición X = Posición Actual X – Posición Final de X

4. Sobrescribir Canal 2 del RC = PidCalcularPitch(Error de posición X)

5. Error de posición Z = Posición Actual Z – Posición Final de Z

6. Sobrescribir Canal 1 del RC = PidCalcularPitch(Error de posición Z)

7. Error de posición Y = Posición Actual Y – Posición Final deY

8. Sobrescribir Canal 3 del RC = PidCalcularRoll(Error de posición Y)

9. Publicar valores sobrescritos en la controladora mediante MavLink

Resumen Algoritmo propuesto

Para una mejor comprensión del controlador que se ha implementado en código C++, se ha desarrollado este capítulo, que a su vez se ha dividido en dos partes:

En primer lugar se muestra el Flow-chart del funcionamiento del controlador en los tres ejes, sobrescribiendo los comandos del Pitch, Roll y Throttle comprendidos entre 1000 y 2000, se logra una aproximación progresiva hasta el punto de aterrizaje.

Posteriormente se ha desarrollado el pseudocódigo de las funciones necesarias para calcular los comandos que han de sacar los controladores a partir de la resta entre la posición actual y la posición final del Quadcopter. En el pseudocódigo hay dos tipos de funciones, las que devuelven un valor int (pidCalcularPitch , pidCalcularThrottle, pidCalcularRoll), que son los controladores devolviendo un comando de esfuerzo, y las que no devuelven nada, (positionCallBack), que se encarga de recibir la posición local del Drone mediante un topic de mavros.

Capítulo 5

V. EXPERIMENTOS Y RESULTADOS

Plataforma de trabajo del UAV

Para conseguir la implantación del controlador PID de alto nivel, este proyecto se apoya en diferentes plataformas de Hardware y Software. En esta sección trataremos de describir cada una de ellas con sus características además de explicar cada una de sus funciones en el proyecto. También se muestran esquemas y figuras de conexionado para su mejor comprensión.

1. MAVROS

MAVROS es un nodo de ROS que proporciona un driver de comunicación desde la controladora de piloto automático mediante el protocolo de comunicación MAVLink hasta una estación de control. El paquete MAVROS incluye las siguientes características [26]:

- Comunicación con el piloto automático mediante un puerto serial.
- Proxy UDP, para la estación de control en Tierra.
- Tópicos compatibles con MAVLink_ros(Mavlink.msg).
- Sistema de 'plugins' (aplicación que se relaciona con otra para agregarle una función específica nueva [28]) para ROS-MAVLink.
- Herramienta de manipulación de parámetros.
- Herramienta de manipulación de puntos de interés.

En este proyecto MAVROS, lo utilizaremos principalmente para:

- Subscribirnos a un 'topic' que nos lea la posición actual del Drone que proviene vía MAVLink de la Controladora PX4.
- Publicar los datos procesados a partir del controlador PID en un 'topic' denominado 'RCOverride', el cual nos permite sobrescribir los valores de los comandos del RC para que este haga una aproximación a la plataforma.

Para conocer todas las funciones que puede realizar MAVROS, visite la página: http://wiki.ros.org/mavros.

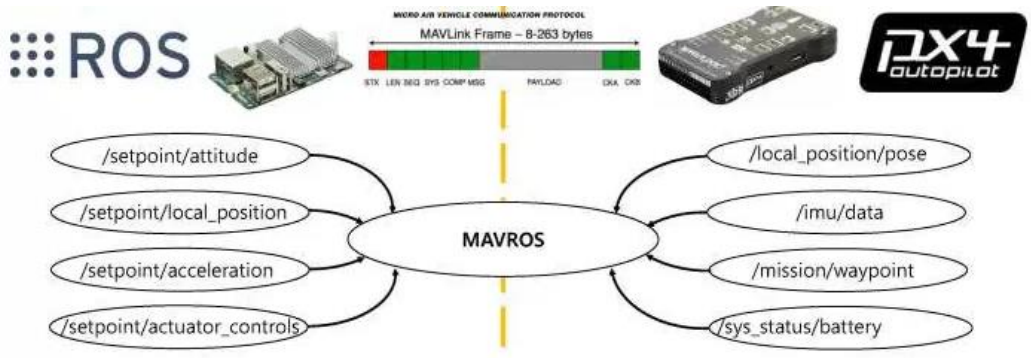

Figura V.1 Esquema de MAVROS [29].

Como se puede observar en la figura anterior, MAVROS es el Nodo que toma la función de nexo entre el sistema operativo del Robot Quadcopter, donde se programan los procesos y la controladora PX4, de donde se reciben los datos de comportamiento del robot.

2. Arquitectura ROS

ROS o sistema operativo robot por sus siglas en inglés, es una plataforma flexible para implementar código de programación a un soporte robot. Es una colección de herramientas y librerías con el objetivo de simplificar la acción de implementar un comportamiento robusto, en nuestro caso, del Quadcopter [25].

El sistema ROS utiliza una tecnología que permite interconectar procesos en el tiempo. Estos procesos se llaman Nodos y la información entre nodos se denomina mensajes. Los mensajes tienen una sola dirección y van a través de un tópico, que es el tema en el que se manda el mensaje. ROS posee una gran variedad de mensajes, desde mensajes primitivos de C++ como son “int” (entero), “doublé” (coma flotante) , “String” (cadena de caracteres) … Hasta librerías como puede ser geometry_msgs que contienen arrays de mensajes primitivos las cuales facilitan mostrar el comportamiento ya sea de velocidad, posición y etc. Del Quadcopter en un momento determinado.

Los Nodos como hemos dicho anteriormente son ejecutables, es decir, en nuestro caso, archivos C++, estos pueden ser “Publishers” o “Subscribers”. Los “Publishers” son aquellos que envían un mensaje

mediante un tópico determinado a otro Nodo. Los "Subscribers" son los procesos o ejecutables que reciben mensajes a partir de otro nodo mediante un tópico para realizar alguna acción determinada. Los tópicos son un método de comunicación asíncrono, pero servicio tiene un concepto diferente, es una comunicación entre dos nodos de forma síncrona. A continuación se muestra un esquema de la arquitectura ROS:

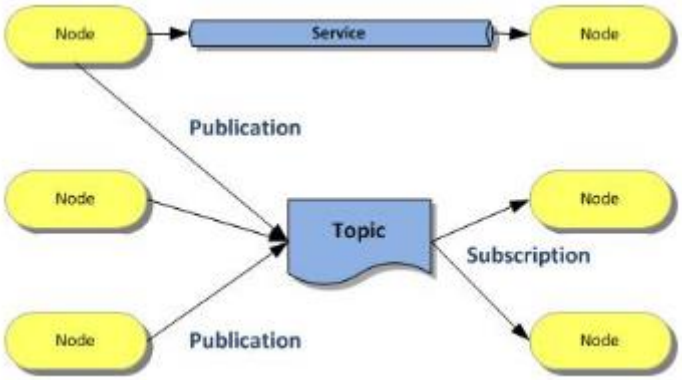

Figura V.2 *Arquitectura ROS [26].*

3. Controladora PixHawk PX4

Pixhawk es un controlador de gran eficiencia y fiabilidad de los datos que analiza e implementa para una determinada trayectoria de un vehículo de plataforma robotizada que tenga un movimiento, como en este caso un Quadcopter. Por esto y por su bajo coste, suele ser muy utilizado en los proyectos de investigación, industria y hobbies. Este controlador que vamos a utilizar funciona con un Procesador Cortex-M4F de 168 MHz/252 MIPS (Microprocesador sin etapas de conductos interbloqueos, por sus siglas en inglés), 2 MB de memoria Flash y tiene 256 KB de RAM [24], para más información vistita la página: https://pixhawk.org/.

Para poder analizar y procesar los datos de comportamiento del Quadcopter, la controladora Pixhawk posee una unidad hardware de sensores que consiste de [14]:

- Un Giróscopo el cual permite conocer la orientación del Drone.
- Magnetómetros y acelerómetros que miden el campo magnético de la tierra al igual que una brújula señala el polo norte magnético.
- Acelerómetros y giróscopos en los tres ejes.

- Barómetro para medir la presión del aire y relacionarlo con la altitud.

En la entrada RC se conecta el receptor del mando control, por donde se introducen los comandos en modo manual, en este proyecto, se cambiará de modo POSITION CONTROL y los comandos se simulan internamente.

Figura V.3 Vista superior y lateral de la controladora [24].

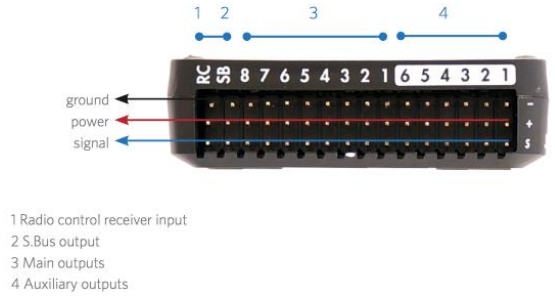

Figura V.4 Vista frontal de la controladora [24].

4. MavLink

MAVLink es un tipo de mensaje de muy poco peso, para ordenar librerías de los UAVS, como en nuestro caso, un Quadcopter. Puede empaquetar estructuras C++ con una alta eficiencia, y sirve como columna vertebral para la comunicación tanto de la IMU/ MCU del Quadcopter como para los interprocesos de la estación de tierra de comunicación en Linux [30].

Como hemos comentado anteriormente MAVROS es un nodo de ROS, que le permite interactúa con el piloto automático mediante un protocolo MAVLink. MAVLink consiste en 17 bytes que incluye la ID del mensaje, la

ID del objetivo y datos. La ID del mensaje muestra en qué consisten los datos que contiene el mensaje [29].

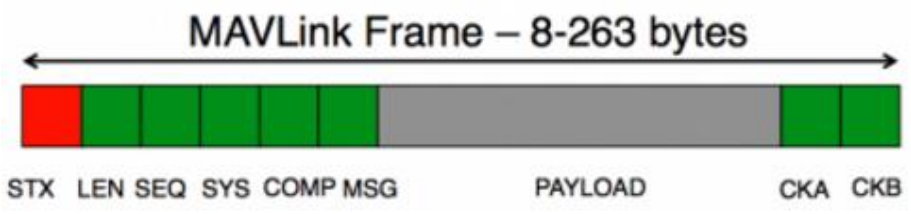

Figura V.5 Estructura MAVLink [29].

Esto permite a MAVLink ser capaz de coger información de múltiples UAVS si el mensaje es transmitido en el mismo canal. Los mensajes pueden ser transmitidos mediante sistemas no cableados, por ejemplo Wi-Fi.

Experimentos Realizados

Una vez desarrollado el algoritmo de control, se tiene que probar su correcto funcionamiento. Se procederá en primer lugar a simulaciones, y posteriormente se implementará el código en el ordenador de a bordo del Quadcopter, para que este haga su correcto funcionamiento.

En esta parte del proyecto se explicara cada uno de las simulaciones que se han realizado además de las pruebas de aproximación reales que ha ejecutado el Quadcopter a un 'Set Point' determinado.

Simulaciones

Test 1. Comandos devueltos por los Controladores desde posiciones aleatorias.

Consiste simular la posición actual del Quadcopter en diferentes puntos de un sistema de coordenadas cartesiano tridimensional. Esta simulación devuelve los comandos del Roll Pitch y Throttle a partir del controlador PID que posteriormente serán los que hagan el 'Over-ride' en la controladora mediante los mensajes MAVLink.

Los puntos en los que se ha realizado la simulación 'Test 1', se muestran en la siguiente gráfica:

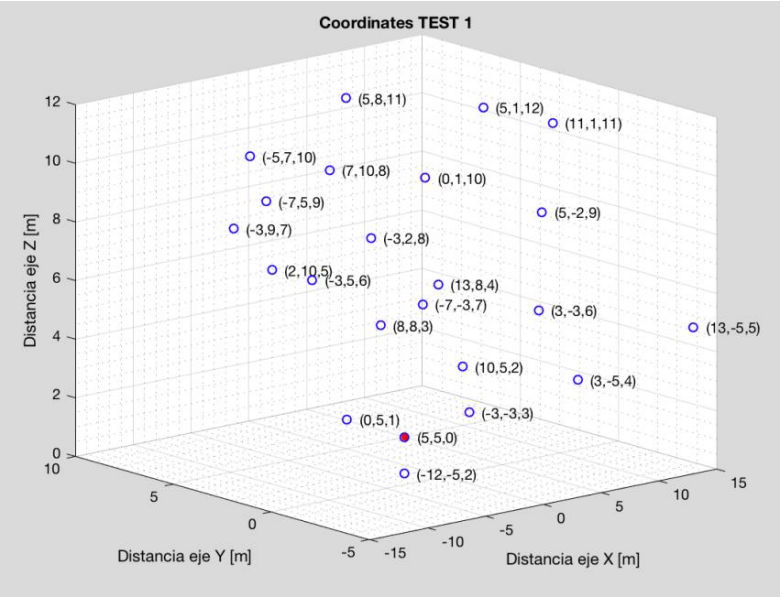

Figura V.6 *Puntos en los que se ha simulado la posición inicial del Quadcopter. Y punto de aterrizaje, en rojo.*

El objetivo de esta prueba es comprobar que se le introduce al Quadcopter valores correctos, y que estos no tengan valores mayores que 2000 o menores que 1000. Esto quiere decir que por norma, la controladora de vuelo solo recibe valores dentro de este rango, por lo tanto en la simulación se intenta recrear al máximo la respuesta del PID realizado. Para ello establecemos las siguientes premisas:

- Las siguientes coordenadas cartesianas corresponden a la posición final Drone (Punto de aterrizaje):

$$X\_final = 5, \qquad Y\_final = 5, \qquad Z\_final = 0$$

- Existen tres controladores en el código: Roll, Pitch y Throttle.
- Los comandos superiores a 1500 en el Roll y Pitch significan que crece el movimiento en el eje positivo de 'Y' y 'X' respectivamente
- Los comandos inferiores a 1500 en el Roll y Pitch informan que decrece el movimiento en los ejes 'Y' y 'X' respectivamente.
- El valor del Throttle en el Test puede variar entre 1000 y 1500. Si se le introduce un valor de 1000 quiere decir que el Quadcopter desciende a la máxima velocidad. Por otro lado, si se le introduce un valor de 1500 es que está en vuelo nivelado a una altitud determinada.
- Dependiendo de la distancia del Quadcopter al punto de aterrizaje, los comandos varían, si el Drone está más lejos, el Roll

y Pitch serán altos, y viceversa. En el caso que el Drone esté situado a mayor altura, recibirá un valor de Throttle menor que si está a poca altura.

- A continuación se muestran las constantes de los controladores que se han utilizado en el Test 1. Como se puede observar, las constantes del Roll y del Pitch, son iguales debido a la simetría de la posición de los motores en el Drone.

PID Roll			*PID Throttle*			*PID Pitch*		
K_p	K_d	K_i	K_{p1}	K_{d1}	K_{i1}	K_{p2}	K_{d2}	K_{i2}
30.00	*10.53*	*12.60*	*20.00*	*15.00*	*10.62*	*30.00*	*10.53*	*12.60*

Tabla V.1 Constantes de los PID's utilizadas en la Prueba 1.

A continuación se mostrará cada uno de los comandos recibidos a partir del PID en el Test 1.

Tabla V.2 *Resultados salida del controlador PID para el eje 'X' (inclinación Pitch).*

	Posición x	***Comando Pitch***
Prueba 0	*0*	*1650*
Prueba 1	*10*	*1350*
Prueba 2	*8*	*1410*
Prueba 3	*13*	*1260*
Prueba 4	*2*	*1590*
Prueba 5	*-3*	*1740*
Prueba 6	*-3*	*1740*
Prueba 7	*7*	*1440*
Prueba 8	*-7*	*1850*
Prueba 9	*-5*	*1800*
Prueba 10	*5*	*1500*
Prueba 11	*5*	*1500*
Prueba 12	*11*	*1320*
Prueba 13	*0*	*1650*
Prueba 14	*5*	*1500*
Prueba 15	*-3*	*1740*
Prueba 16	*-7*	*1860*
Prueba 17	*3*	*1560*
Prueba 18	*13*	*1260*
Prueba 19	*3*	*1560*
Prueba 20	*-3*	*1740*
Prueba 21	*-12*	*2000*

Tabla V.3 Resultados salida del controlador PID para el eje 'Y' (inclinación Roll).

	Posición y	***Comando Roll***
Prueba 0	*5*	*1500*
Prueba 1	*5*	*1500*
Prueba 2	*8*	*1410*
Prueba 3	*8*	*1410*
Prueba 4	*10*	*1350*
Prueba 5	*5*	*1500*
Prueba 6	*9*	*1380*
Prueba 7	*10*	*1350*
Prueba 8	*5*	*1500*
Prueba 9	*7*	*1440*
Prueba 10	*8*	*1410*
Prueba 11	*1*	*1620*
Prueba 12	*1*	*1620*
Prueba 13	*1*	*1620*
Prueba 14	*-2*	*1710*
Prueba 15	*2*	*1590*
Prueba 16	*-3*	*1740*
Prueba 17	*-3*	*1740*
Prueba 18	*-5*	*1800*
Prueba 19	*-5*	*1800*
Prueba 20	*-3*	*1740*
Prueba 21	*-5*	*1800*

Tabla V.4 Resultados salida del controlador PID para el eje 'Z' (Empuje).

	Posición z	***Comando Throttle***
Prueba 0	*1*	*1480*
Prueba 1	*2*	*1460*
Prueba 2	*3*	*1440*
Prueba 3	*4*	*1420*
Prueba 4	*5*	*1400*
Prueba 5	*6*	*1380*
Prueba 6	*7*	*1360*
Prueba 7	*8*	*1340*
Prueba 8	*9*	*1320*
Prueba 9	*10*	*1300*
Prueba 10	*11*	*1280*
Prueba 11	*12*	*1260*
Prueba 12	*11*	*1280*
Prueba 13	*10*	*1300*
Prueba 14	*9*	*1320*
Prueba 15	*8*	*1340*
Prueba 16	*7*	*1360*
Prueba 17	*6*	*1380*
Prueba 18	*5*	*1400*
Prueba 19	*4*	*1420*
Prueba 20	*3*	*1440*
Prueba 21	*2*	*1460*

Tabla V.5 Resultados Totales del Test 1 combinados.

Test día 5/Mayo/2017	***Posición del Drone***			***Comandos del Drone***		
	x	***y***	***z***	***Pitch***	***Roll***	***Throttle***
Prueba 0	*0*	*5*	*1*	*1650*	*1500*	*1480*
Prueba 1	*10*	*5*	*2*	*1350*	*1500*	*1460*
Prueba 2	*8*	*8*	*3*	*1410*	*1410*	*1440*
Prueba 3	*13*	*8*	*4*	*1260*	*1410*	*1420*
Prueba 4	*2*	*10*	*5*	*1590*	*1350*	*1400*
Prueba 5	*-3*	*5*	*6*	*1740*	*1500*	*1380*
Prueba 6	*-3*	*9*	*7*	*1740*	*1380*	*1360*
Prueba 7	*7*	*10*	*8*	*1440*	*1350*	*1340*
Prueba 8	*-7*	*5*	*9*	*1850*	*1500*	*1320*
Prueba 9	*-5*	*7*	*10*	*1800*	*1440*	*1300*
Prueba 10	*5*	*8*	*11*	*1500*	*1410*	*1280*
Prueba 11	*5*	*1*	*12*	*1500*	*1620*	*1260*
Prueba 12	*11*	*1*	*11*	*1320*	*1620*	*1280*
Prueba 13	*0*	*1*	*10*	*1650*	*1620*	*1300*
Prueba 14	*5*	*-2*	*9*	*1500*	*1710*	*1320*
Prueba 15	*-3*	*2*	*8*	*1740*	*1590*	*1340*
Prueba 16	*-7*	*-3*	*7*	*1860*	*1740*	*1360*
Prueba 17	*3*	*-3*	*6*	*1560*	*1740*	*1380*
Prueba 18	*13*	*-5*	*5*	*1260*	*1800*	*1400*
Prueba 19	*3*	*-5*	*4*	*1560*	*1800*	*1420*
Prueba 20	*-3*	*-3*	*3*	*1740*	*1740*	*1440*
Prueba 21	*-12*	*-5*	*2*	*2000*	*1800*	*1460*

Como se puede observar los valores obtenidos están dentro del rango requerido, y cumplen las premisas anteriores. Por lo tanto a los comandos de la aproximación que le introducimos a la controladora de vuelo se les validan parcialmente, ya que se puede variar en pruebas posteriores:

- Las constantes del PID para cada uno de los movimientos, debido a una mejor eficiencia de vuelo en las pruebas reales.
- Valor de la calibración del vuelo estable (vuelo estacionario). En el 'Test 1', se ha establecido el valor medio entre 1000 y 2000 (1500), como el comando que envía la señal para mantenerse estable en el eje, ya sea Pitch como Roll. Este valor puede variar unas décimas y situarse en posiciones como 1490, 1450,1610, etc. Pero siempre cerca del 1500.

Una vez simulado el controlador PID, se obtienen los datos de la tabla anterior. Los comandos del pitch y Roll y Throttle se vectorizan y se obtienen las siguientes gráficas, las cuales muestran los vectores de la aproximación dependiendo del punto dónde se encuentre el Quadcopter.

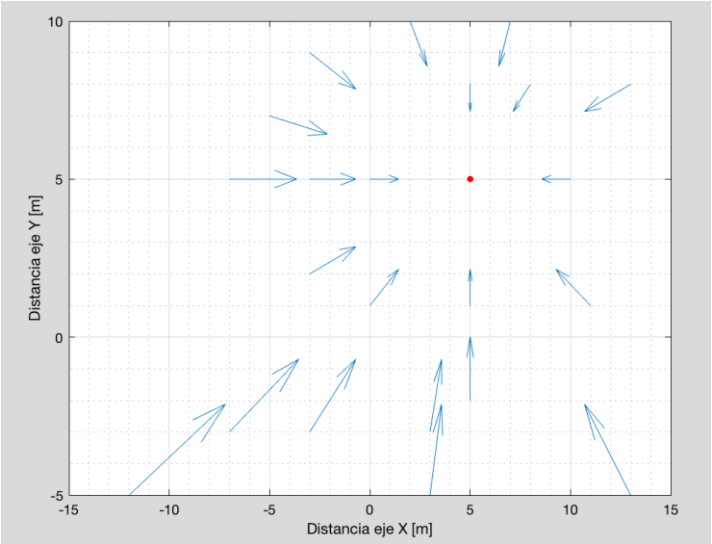

Figura V.7 Vectores Pitch y Roll, Aproximándose al punto de aterrizaje.

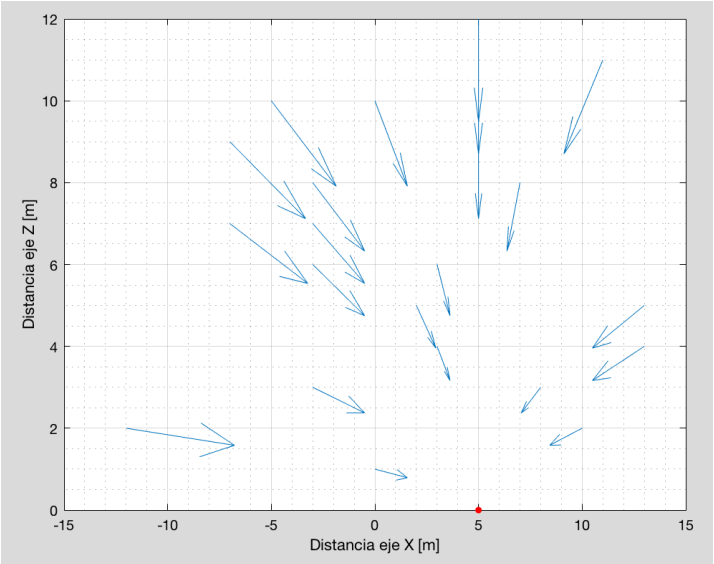

Figura V.8 Vectores Pitch y Throttle Aproximándose al punto de aterrizaje.

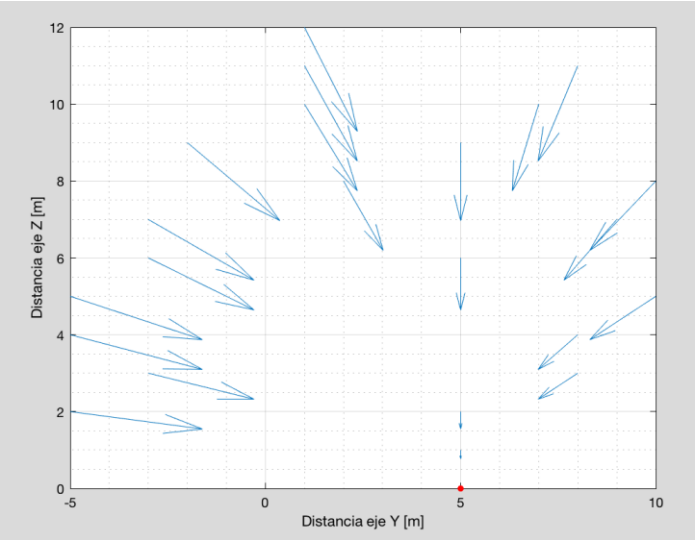

Figura V.9 Vectores Roll y Throttle, Aproximándose al punto de aterrizaje.

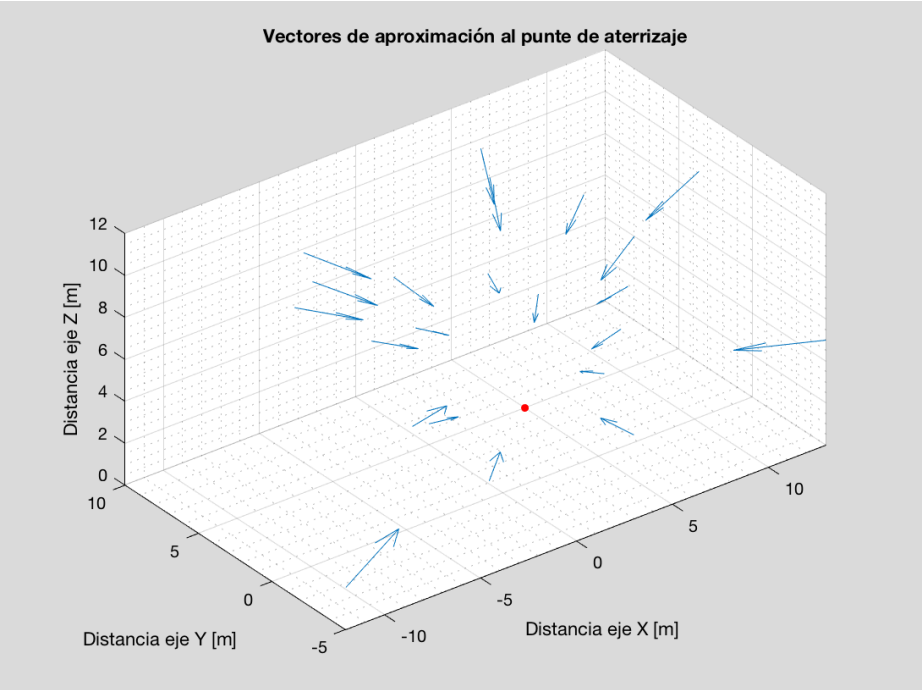

Figura V.10 Vectores Roll, Pitch y Throttle, Aproximándose al punto de aterrizaje.

Se llega a la conclusión que los controladores PID creados en el código de C++ computan correctamente los datos, lo que significa que únicamente habría que variar las constates de los PID para modificar el comportamiento del Drone (mayor rapidez en cambio de movimientos, o más lento para un vuelo más seguro) según los requerimientos del cliente.

Test 2. Simulación de trayectorias simples

Esta segunda simulación consiste en analizar la respuesta de los comandos Roll Pitch y Throttle devueltos por los controladores ante una trayectoria simple. Para comprobar cada comando devuelto por el controlador por separado, se simularán tres trayectorias distintas, cada una de las cuales es una sucesión de puntos alineados paralelas a cada eje de coordenadas. Las características de la simulación 'Test 2' son las siguientes:

- El programa irá leyendo una sucesión de coordenadas que anteriormente se han introducido en un fichero.
- Las trayectorias para comprobar el comando devuelto Pitch y Roll se encontrarán en el plano X-Y, estando cada uno de las coordenadas a la misma altura y formando en conjunto una línea recta paralela a el eje X (para comprobar el comando Pitch), o al eje Y (comando Roll).
- La trayectoria que se simula para analizar el comando del Throttle devuelto, consiste en una sucesión de coordenadas sin variación en los ejes X ó Y, siendo la alteración en el eje Z.
- Las coordenadas para el análisis del Pitch comienzan en X=0, y terminan en X=17.
- Las coordenadas para el análisis del Roll comienzan en Y=0, y terminan en Y=17.
- Las coordenadas para el análisis del Throttle comienzan en Z=17, disminuyen hasta Z=5, y vuelven a aumentar para llegar finalmente a Z=17.
- Se ha realizado un diseño de trayectorias en el cual la distancia entre puntos dentro de las mismas, va disminuyendo progresivamente, esto se ha realizado para poder analizar la respuesta de los controladores ante distintas distancias.
- El comando devuelto estará en un rango de 1000 a 2000 siendo 1500 posición de equilibrio. Si el Comando es mayor que 1500, el Drone se desplaza en dirección positiva al eje X si el comando es Pitch, Y, para el comando Roll, y Z para el comando Throttle.
- Las constantes de los controladores son las mismas utilizadas que en el Test 1:

- **Tabla V.6** Constantes de os controladores para el Test 2.

PID Roll			*PID Throttle*			*PID Pitch*		
K_p	K_d	K_i	K_{p1}	K_{d1}	K_{i1}	K_{p2}	K_{d2}	K_{i2}
30.00	*10.53*	*12.60*	*20.00*	*15.00*	*10.62*	*30.00*	*10.53*	*12.60*

Las gráficas que se han simulado en los controladores se muestran a continuación en la Figura 5.11, 5.12, y 5.13.

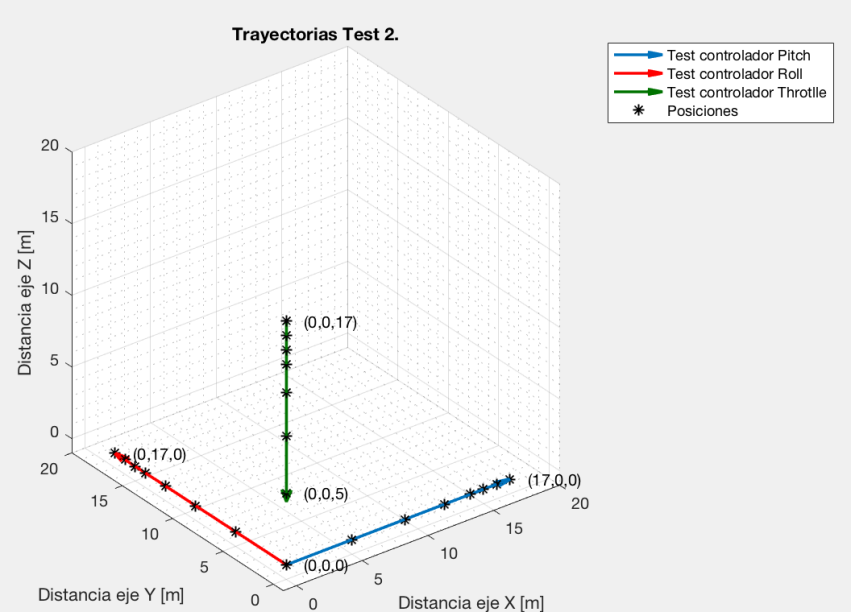

Figura V.11 Simulación trayectoria recta para observar correcto funcionamiento del Throttle.

Una vez que se han simulado las trayectorias anteriores, se han obtenido las siguientes tablas de resultados, en la que se puede ver la respuesta de cada controlador a la trayectoria ejecutada:

Tabla V.7 Resultados Comando Pitch

	Coordenadas			Comando
Position	**x**	**y**	**z**	**Pitch**
1	0	0	0	1650
2	5	0	0	1607
3	9	0	0	1577
4	12	0	0	1547
5	14	0	0	1517
6	15	0	0	1530
7	16	0	0	1530
8	17	0	0	1500

Tabla V.8 Resultados Comando Roll.

	Coordenadas			Comando
Position	**x**	**y**	**z**	**Roll**
1	0	0	0	1650
2	0	5	0	1607
3	0	9	0	1577
4	0	12	0	1547
5	0	14	0	1517
6	0	15	0	1530
7	0	16	0	1530
8	0	17	0	1500

Tabla V.9 Resultado comando Throttle.

	Coordenadas			Comando
Position	**x**	**y**	**z**	**Throttle**
1	0	0	17	1480
2	0	0	16	1480
3	0	0	15	1480
4	0	0	14	1480
5	0	0	12	1450
6	0	0	9	1430
7	0	0	5	1410
8	0	0	9	1529
9	0	0	12	1509
10	0	0	14	1519
11	0	0	16	1519
12	0	0	17	1519

Los resultados obtenidos en la simulación mostradros en la Tabla 5.5, 5.6, y 5.7, se representan a continuación:

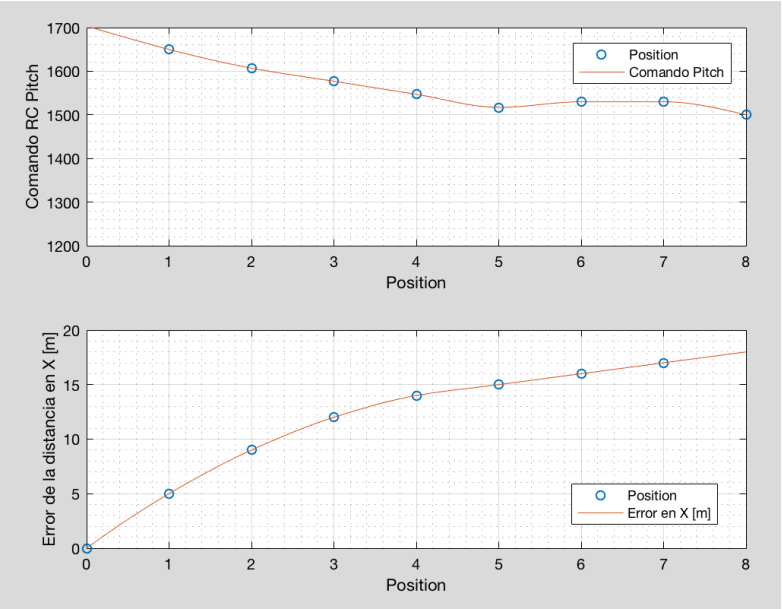

Figura V.12 Respuesta del PID en comando Pitch ante el error en el eje 'X' de la trayectoria simulada 2.1.

En la figura 5.14 se muestran dos sub gráficas, en la inferior se puede apreciar la distancia del Drone a medida que se va desplazando por la trayectoria paralela al eje X desde una posición inicial (X=0), hasta una posición final (X=17). Por otro lado la gráfica superior representa el valor del Comando del Pitch devuelto a medida que va pasando por las diferentes posiciones de la trayectoria. Se aprecia que el comando del Pitch tiene valores altos en las posiciones iniciales, en el que la variación del Error en X (distancia), entre posiciones es mayor. Esto quiere decir que el controlador devuelve valores mayores para que el cambio entre posiciones sea más rápido si las distancias son más grandes. Por otro lado se observa una sobreoscilación en la posición 7, debido al cambio de tendencia en la variación entre posiciones siendo esta menor a partir de la posición 7.

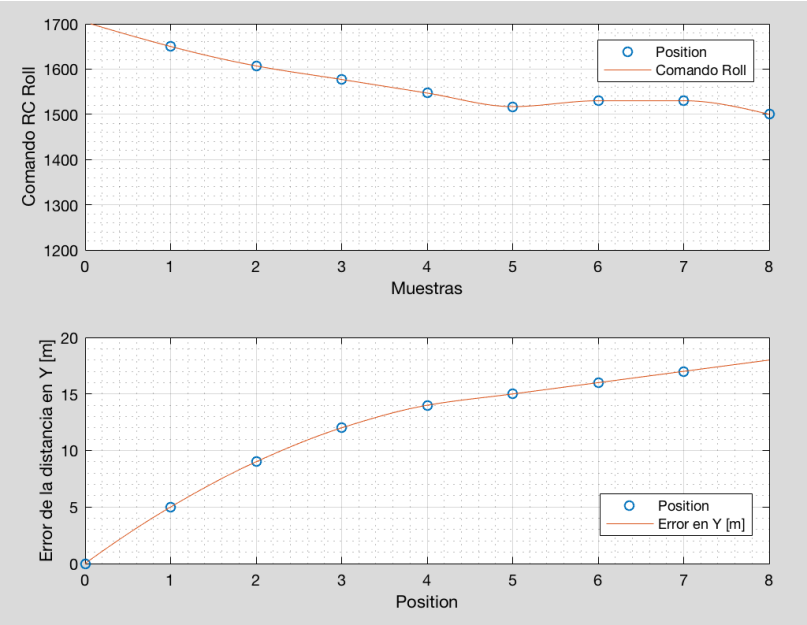

Figura V.13 Respuesta del PID en comando Pitch ante el error en el eje 'X' de la trayectoria simulada 2.1.

En la figura 5.15 se observa exactamente lo mismo que en la Figura 5.16, esto se debe a que se ha simulado la misma variación de posiciones tanto en el eje X como en el eje Y, para conocer el correcto funcionamiento del comando Roll. Los valores son idénticos porque las constantes de los controladores de Pitch y Roll son las mismas debido a la simetría del Drone.

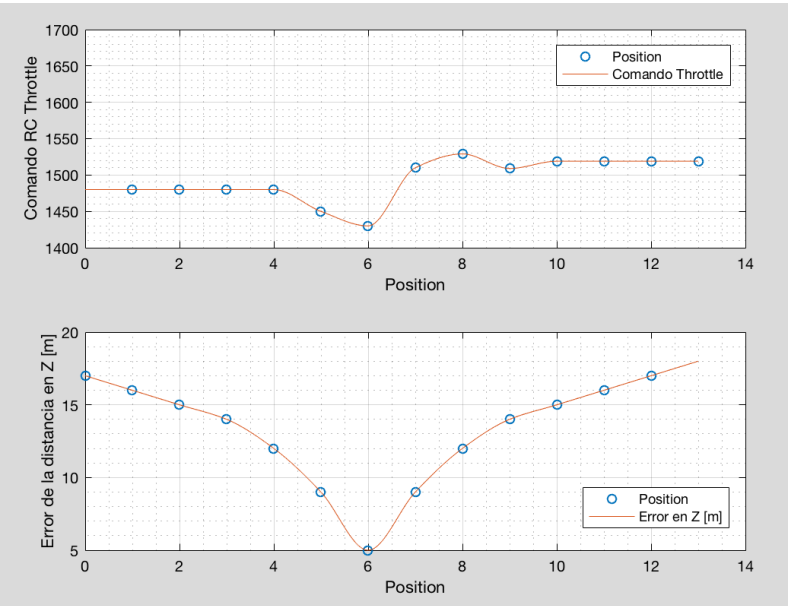

Figura V.14 Respuesta del PID en comando Pitch ante el error en el eje 'X' de la trayectoria simulada 2.1.

En la figura 5.16 se observa la respuesta del controlador del Throttle a medida que se varía la altitud simulada del Drone. Como se ha comentado anteriormente, el Drone parte de una posición inicial Z=17 y llega hasta un mínimo para Z=5, posteriormente ha de volver a elevarse hasta la posición inicial.

Desde la posición 0 hasta la posición número 6 el Drone desciende progresivamente. Como se puede observar en el sub-gráfico superior de la figura el comando del Throttle es ligeramente inferior a 1500. De la posición 4 a la posición 6 aumenta la diferencia del alturas entre posiciones de la trayectorias, lo que significa que la velocidad de descenso ha de ser mayor, por lo tanto el comando disminuye hasta un valor de 1410 en la posición 6, lo cual, concuerda con la lógica de funcionamiento.

Desde la posición 6 hasta la 12 se realiza el ascenso desde Z=5 hasta Z=17, por lo tanto los valores del Throttle han de ser superiores a 1500 concordando con los resultados de la simulación. Existe una sobreoscilación del controlador en la posición 8 debido al cambio de progresión en la velocidad de ascenso.

Test 3. ‘Dynamic’

Esta segunda prueba ha ido enfocada a comprobar el comportamiento de los controladores del Drone ante una sucesión de coordenadas para un aterrizaje seguro en un punto determinado. Las coordenadas de aproximación se introducen en un archivo de texto ‘comma-separated values’, de la misma manera que se ha hecho en el Test 2, El programa convierte esas coordenadas en enteros y las va introduciendo una a una en los controladores mediante un bucle. Para esta simulación se establecen las siguientes premisas:

- Los comandos devueltos por los controladores del Pitch Roll y Throttle varían entre 1000 y 2000 (como en el Test 1), siendo valores superiores a 1500 como movimientos positivos en el eje de las X, Y, y Z respectivamente.

- Se simularán dos trayectorias, ambas empezarán en el Punto (0, 0, 10) e irán descendiendo hacia el (0, 0, 0), mediante las aproximaciones. Estas trayectorias de aproximación se muestran posteriormente.

- Las trayectorias las adquiere el programa mediante la lectura de un archivo.

- Las constantes de los controladores son las mismas utilizadas que en el Test 1:

Tabla V.10 Constantes de los PID's utilizadas en la Prueba 3.

PID Roll			*PID Throttle*			*PID Pitch*		
K_p	K_d	K_i	K_{p1}	K_{d1}	K_{i1}	K_{p2}	K_{d2}	K_{i2}
30.00	*10.53*	*12.60*	*20.00*	*15.00*	*10.62*	*30.00*	*10.53*	*12.60*

Trayectoria 3.1

En primer lugar se han simulado los puntos por los que el Drone tiene que pasar

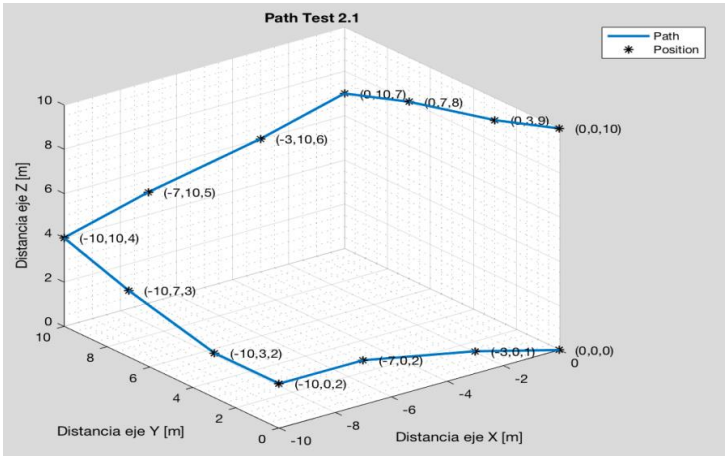

Figura V.15 Trayectoria 2.1. Simulada.

Los resultados de la trayectoria 3.1 se muestran en las tablas siguientes:

Tabla V.11 Comandos devueltos por el Controlador ante la trayectoria 3.1.

Resultados Test 2. 'Dynamic'
Test día 14 de Mayo 2017

	Posición Inicial			***Posición Final***			***Comandos***		
STEP	*X*	*Y*	*Z*	*Xf*	*Yf*	*Zf*	***Pitch***	***Roll***	***Throttle***
1	*0*	*0*	*10*	*0*	*3*	*9*	*1500*	*1590*	*1480*
2	*0*	*3*	*9*	*0*	*7*	*8*	*1500*	*1632*	*1480*
3	*0*	*7*	*8*	*0*	*10*	*7*	*1500*	*1463*	*1480*
4	*0*	*10*	*7*	*-3*	*10*	*6*	*1368*	*1500*	*1480*
5	*-3*	*10*	*6*	*-7*	*10*	*5*	*1368*	*1500*	*1480*
6	*-7*	*10*	*5*	*-10*	*10*	*4*	*1423*	*1500*	*1480*
7	*-10*	*10*	*4*	*-10*	*7*	*3*	*1537*	*1373*	*1480*
8	*-10*	*7*	*3*	*-10*	*3*	*2*	*1500*	*1368*	*1480*
9	*-10*	*3*	*2*	*-10*	*0*	*2*	*1500*	*1423*	*1510*
10	*-10*	*0*	*2*	*-7*	*0*	*2*	*1627*	*1538*	*1500*
11	*-7*	*0*	*2*	*-3*	*0*	*1*	*1632*	*1500*	*1470*
12	*-3*	*0*	*1*	*0*	*0*	*0*	*1577*	*1500*	*1480*

Una vez obtenidos los comandos del RC para cada uno de los ejes, se puede representar la respuesta del PID ante la variación del error de distancia. Los resultados graficados se muestran a continuación:

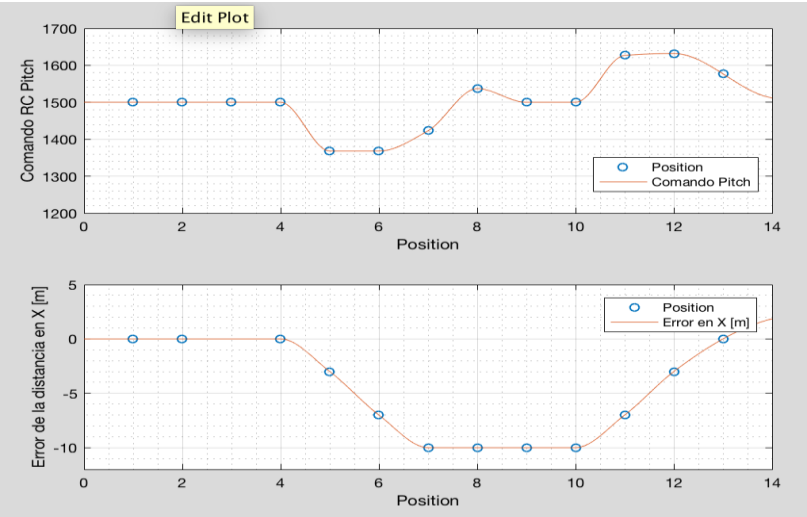

Figura V.16 Respuesta del PID en comando Pitch ante el error en el eje 'X' de la trayectoria simulada 2.1.

En esta gráfica se observa el cambio del comando del Pitch a medida que variamos el error de la distancia del eje X. Cuando el valor permanece constante el controlador del Pitch no varía. En el momento que el Drone comienza a moverse a lo largo del eje X (Muestra 4), el controlador devuelve un comando determinado entre los valores 1000 y 2000. Si el movimiento es positivo en el eje, este comando será superior a 1500 e inferior a 2000 (Muestra 10), en cambio si el movimiento es negativo, el comando será inferior a 1500 y superior a 1000. También se puede apreciar la sobreosciolación (Muestra 8) que se produce al cambiar la tendencia del error de posición. Esta sobreosciolación se puede minimizar cambiando los valores de las constantes de los controladores PID.

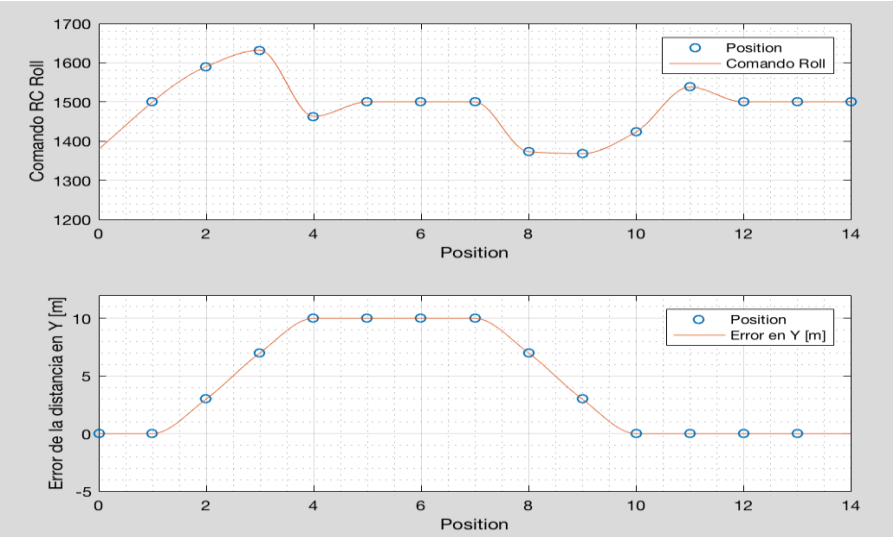

Figura V.17 Respuesta del PID en comando Roll ante el error en el eje 'Y' de la trayectoria simulada 2.1.

Como ya se ha comentado anteriormente, al ser un Drone simétrico, las constantes de los controladores del Roll y del Pitch son las mismas por lo tanto las variaciones idénticas en el error de la distancia en los ejes X, e Y, dan una respuesta prácticamente idéntica de los comandos del Pitch y Roll. En la Figura 5.19 se puede apreciar también una sobreosciolación en la muestra 4 debido al cambio de tendencia del error. También observamos que si el error de la distancia permanece constante, el comando devuelto por el PID también permanece constante a un valor determinado.

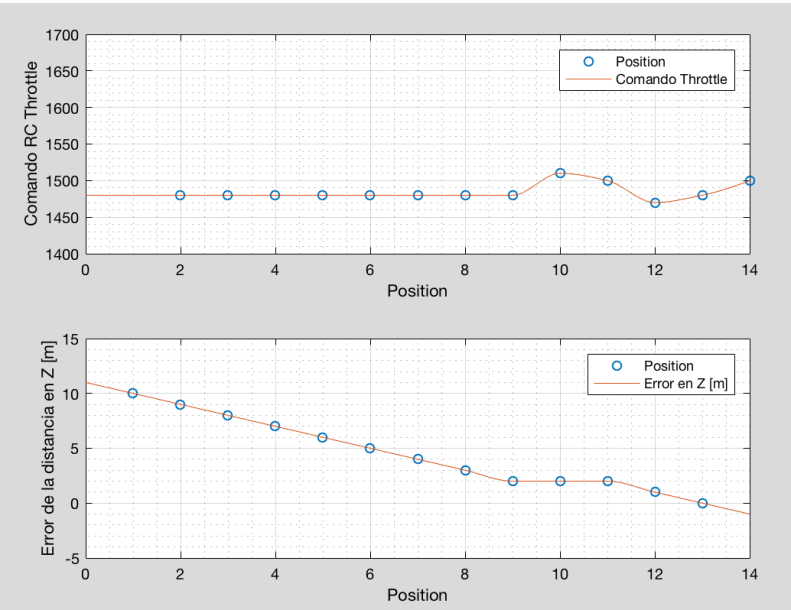

Figura V.18 Respuesta del PID en comando Throttle ante el error en el eje 'Z' de la trayectoria simulada 2.1.

Las constantes del controlador PID del Throttle son distintas que las de los PID's del Pitch y Roll por lo tanto se comporta de una manera ligeramente diferente. En esta última gráfica se observa que hay un descenso de altitud en Z progresivo desde 10 metros (Muestra 1) hasta 2 metros (Muestra 9), este descenso el controlador lo interpreta como un valor ligeramente inferior a 1500, exactamente 1480. En la muestra 9 hay un periodo de altitud estable hasta la muestra 11, En el controlador aparece una pequeña sobreosciolación que alcanza el nivel de 1510 antes de estabilizarse posteriormente en el valor de equilibrio de 1500. A partir de la muestra 11 vuelve a descender desde los 2 metros hasta los cero metros para completar su aterrizaje. Esta etapa el PID la interpreta con un valor de 1470, inferior a 1500 por lo tanto, disminuye altitud.

Trayectoria 3.2

Una vez analizados los datos de la trayectoria 3.1 pasaremos a estudiar los datos de la trayectoria 3.2. Esta es más completa, se compone de 20 puntos distribuidos en el espacio por los que el Drone ha de pasar. Esta trayectoria consiste en mover el Drone por un recorrido en forma de eneágono irregular en vista de planta (X, Y), y a su vez realizando un descenso desde los 10 metros hasta 0, con trayectos intermedios en los que la altitud permanece constante. El fin de esta simulación es exponer a los reguladores a múltiples cambios de dirección en cortos periodos de tiempo para medir las sobreoscilaciones de los controladores Pitch, Roll y Throttle. El recorrido 3.2 simulado es el siguiente:

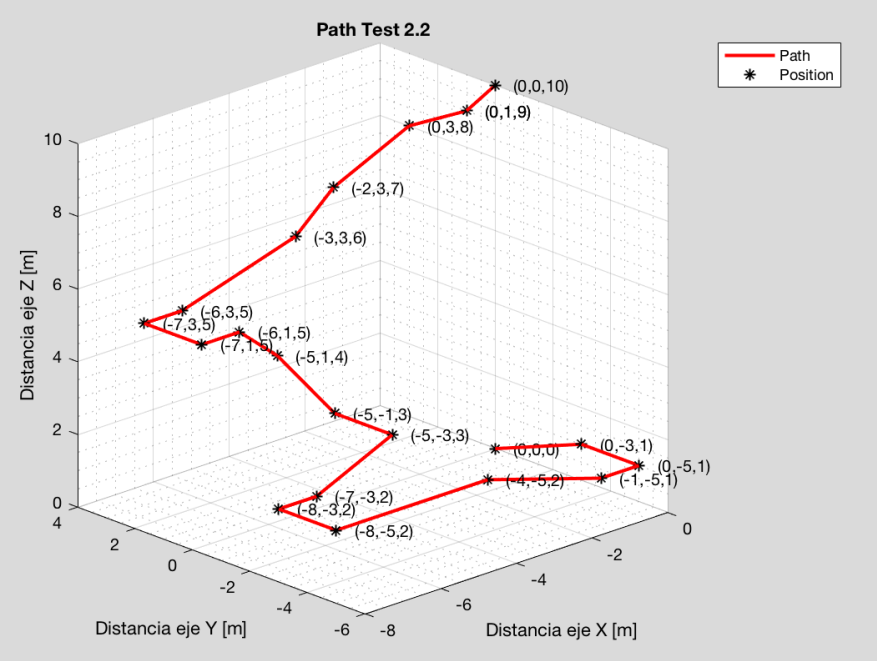

Figura V.19 Trayectoria 2.2. Simulada.

Una vez obtenidos los datos anteriores, se puede formar la tabla siguiente, la cual posee todos los datos de la trayectoria 3.2. Unidos:

Tabla V.12 Resultados de la simulación Trayectoria 3.2.

Resultados Test 2.2 'Dynamic' Test día 17 de Mayo 2017

	Posición Inicial			***Posición Final***			***Comandos***		
STEP	*X*	*Y*	*Z*	*Xf*	*Yf*	*Zf*	***Pitch***	***Roll***	***Throttle***
1	*0*	*0*	*10*	*0*	*1*	*9*	*1500*	*1530*	*1480*
2	*0*	*1*	*9*	*0*	*3*	*8*	*1500*	*1572*	*1480*
3	*0*	*3*	*8*	*-2*	*3*	*7*	*1415*	*1476*	*1480*
4	*-2*	*3*	*7*	*-3*	*3*	*6*	*1483*	*1500*	*1480*
5	*-3*	*3*	*6*	*-6*	*3*	*5*	*1385*	*1500*	*1480*
6	*-6*	*3*	*5*	*-7*	*3*	*5*	*1495*	*1500*	*1510*
7	*-7*	*3*	*5*	*-7*	*1*	*5*	*1512*	*1416*	*1500*
8	*-7*	*1*	*5*	*-6*	*1*	*5*	*1542*	*1525*	*1500*
9	*-6*	*1*	*5*	*-5*	*1*	*4*	*1529*	*1500*	*1470*
10	*-5*	*1*	*4*	*-5*	*-1*	*3*	*1488*	*1416*	*1480*
11	*-5*	*-1*	*3*	*-5*	*-3*	*3*	*1500*	*1441*	*1510*
12	*-5*	*-3*	*3*	*-7*	*-3*	*2*	*1415*	*1525*	*1470*
13	*-7*	*-3*	*2*	*-8*	*-3*	*2*	*1483*	*1500*	*1510*
14	*-8*	*-3*	*2*	*-8*	*-5*	*2*	*1512*	*1416*	*1500*
15	*-8*	*-5*	*2*	*-4*	*-5*	*2*	*1670*	*1525*	*1500*
16	*-4*	*-5*	*2*	*-1*	*-5*	*1*	*1577*	*1500*	*1470*
17	*-1*	*-5*	*1*	*0*	*-5*	*1*	*1504*	*1500*	*1510*
18	*0*	*-5*	*1*	*0*	*-3*	*1*	*1488*	*1585*	*1500*
19	*0*	*-3*	*1*	*0*	*0*	*0*	*1500*	*1602*	*1470*

Los resultados de la trayectoria 3.2 se muestran graficados en las Figura 5.22, 5.23 y 5.24:

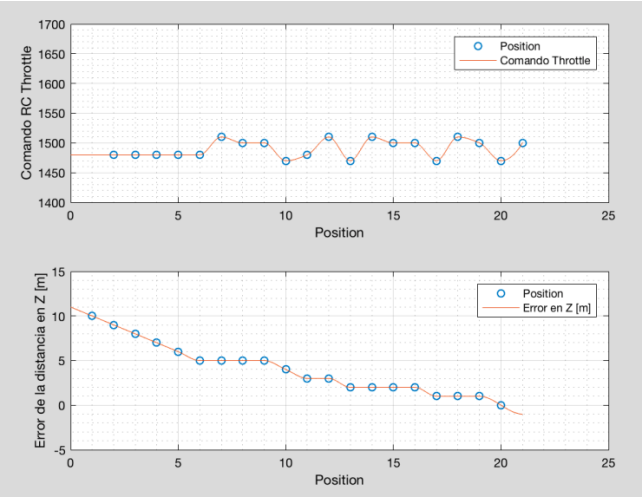

Figura V.20 Respuesta del PID en comando Throttle ante el error en el eje 'Z' de la trayectoria simulada 3.2.

En la figura 5.22 se aprecia la respuesta del comando RC del Throttle ante el cambio en altitud del Drone. La altitud de la trayectoria 3.2 parte desde una altura Z=10 y llega a la posición final de Z=0. Se puede apreciar en el sub-gráfico inferior de la figura, que existen trayectos intermedios en la trayectoria en los que la altura permanece constante, esto se ha simulado de esta para ver el cambio de tendencia de la respuesta del controlador del comando RC del Throttle.

Desde la posición Z=10 hasta Z=5 el régimen de descenso del Drone permanece constante lo que se traduce a que el comando ha de ser ligeramente inferior a 1500. Como se observa en el sub-gráfico superior de la figura esto se cumple. Posteriormente la altitud del Drone permanece constante durante una parte de la trayectoria, posiciones 5,6,7,8 y 9, por lo que el comando ha de volver a una posición de equilibrio (1500). Se aprecia que esto se cumple pero como siempre aparece una pequeña sobreoscilacón cuando el Drone cambia entre dos altura.

La posición número 12 es de interés debido a que existe un cambio rápido de altura, hay un máximo del comando RC 1510 (sobreoscilación) que permite que el Drone vuelva a un régimen de descenso nulo durante un corto periodo de tiempo. Pero rápidamente el Drone ha de descender de nuevo, y el comando tiene que ser inferior a 1500, como se ve en el sub-gráfico superior esto se cumple, con un valor de 1470.

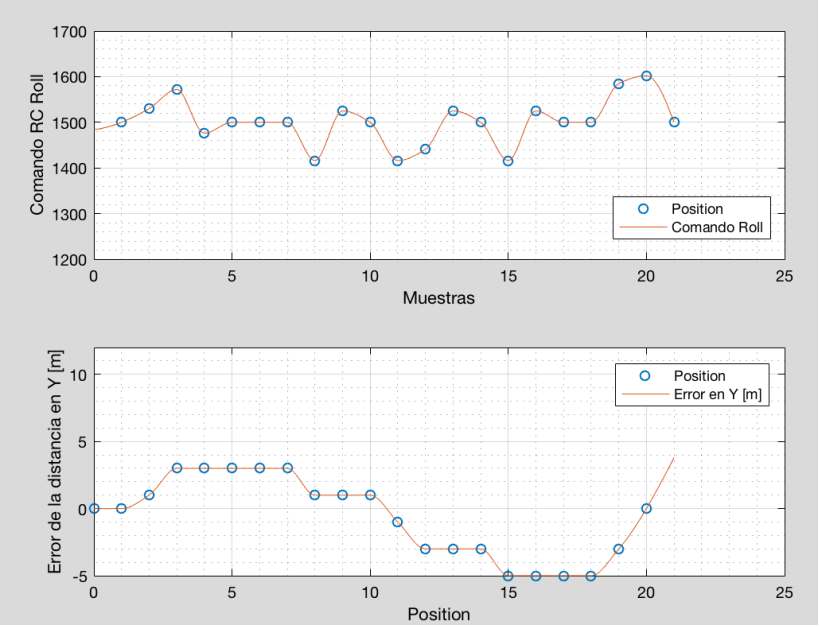

Figura V.21 Respuesta del PID en comando Roll ante el error en el eje 'Y' de la trayectoria simulada 3.2.

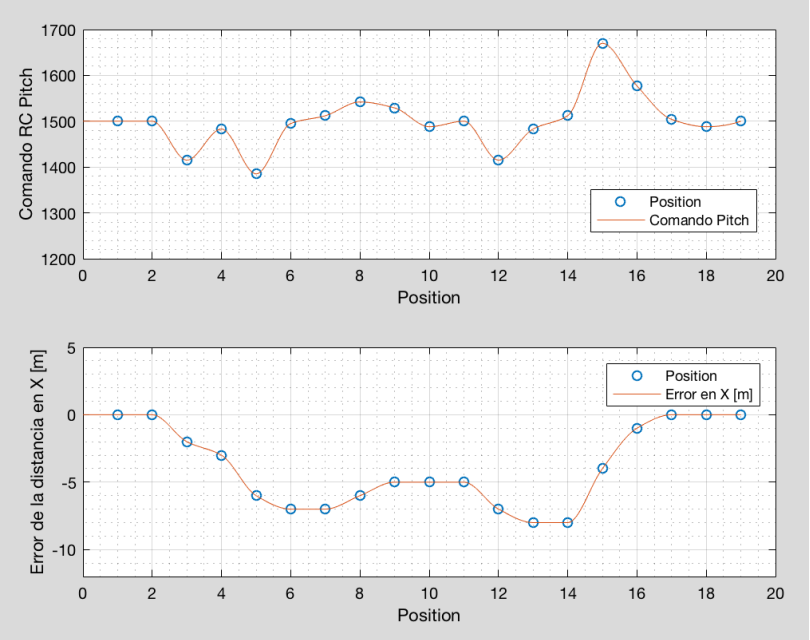

Figura V.22 Respuesta del PID en comando Pitch ante el error en el eje 'X' de la trayectoria simulada 3.2.

Las gráficas de las Figuras 5.23 y 5.24 responden a las respuestas de los comandos de los controladores RC de Pitch y Roll, como se expresa en la Tabla 5.10 ambos tienen las mismas constantes Proporcional, Derivativa e Integral.

En la Figura 5.24 se representa la respuesta del comando Pitch a media que el error en posición en el eje X cambia. Desde la posición 2 a la posición 6 el Drone se aleja en el eje X pero se aprecia que el comando controlador devuelve una sobreoscilación en el punto 4, esto se debe a que el Drone experimenta un cambio en la tendencia de velocidad de desplazamiento en X. Para poder arreglar esta sobreoscilación habría que ajustar las constantes del PID aumentando la constante derivativa, o disminuyendo la integral o proporcional.

En la posición 15 se debe mencionar otra gran sobreoscilación de valor de 1670 porque en este caso el cambio desde la posición X=-7 a X=0 es muy rápido. Como se ha comentado en el párrafo anterior se deberían de cambiar las constantes, especialmente aumentando la derivativa.

Resumen Experimentos y Resultados

Este capítulo se ha dividido principalmente en dos partes, la primera constituye una introducción a los materiales de laboratorio con los que se ha desarrollado el proyecto: MAVROS, ROS, C++, MAVLink, PIXHAWK y Ubuntu 16.04. Se explica su funcionamiento y funciones que se han realizado con ellos.

En la segunda parte se muestran los experimentos realizados con el controlador una vez implementado. Se analizan las salidas del controlador que posteriormente se introducirán a MAVROS para que la controladora de vuelo pueda leerlos.

Los resultados obtenidos a partir de las pruebas se han tabulado, y seguidamente mediante varios programas de MATLAB se han mostrado en gráficos para un mejor análisis y comprensión.

Como se ha comentado anteriormente, cada uno de los resultados graficados se ha analizado y obtenido conclusiones.

Capítulo 6

VI. TRABAJOS FUTUROS Y CONCLUSIONES

Conclusiones

Después de estos 5 meses trabajando en el proyecto se ha logrado implementar un controlador de aterrizaje a un Quadcopter. Este controlador recibe los datos de posición local del Drone mediante mensajes MavLink en coordenadas X, Y, y Z, y mediante una realimentación negativa se restan a la posición final. La salida de los controladores consiste en un comando entre 1000 y 2000 con 1500 como posición de equilibrio que se introduce mediante MavLink a canales de la controladora y esta a su vez calcula cada una de las fuerzas ejercidas a los motores.

El controlador implementado ha sido un PID, va orientado al aterrizar sobre un punto pre-programado. Se puede implementar tanto en una controladora Pixhawk con el Software PX4 como para el Sofware de Ardupilot, uno de los más punteros en la navegación Autónoma. Principalmente se ha tenido en cuenta que el Drone que se ha programado tiene configuración en ´X´, pero también se puede configurar un Quadcopter en configuración ´plus´ con este controlador PID de aterrizaje.

Se han obtenido gran cantidad de resultados para estudiar el comportamiento del controlador. El primer Test consistía en programar un punto de aterrizaje y obtener la tendencia del Drone situándolo en un punto aleatorio del sistema de coordenadas. Posteriormente se pasó a obtener la respuesta de los comandos devueltos por el Controlador para analizar sobreocilaciones o errores de régimen permanente que pudiesen aparecer. Y por último se programaron trayectorias complejas, las cuales el Drone ha recorrido para de nuevo obtener más detalladamente las respuestas de los controladores.

Otro de los campos que también se han explorado es la instalación de Gazebo, un simulador a tiempo real donde se puede comprobar los correctos funcionamientos de los controladores.

Trabajos futuros

El fin de este proyecto siempre ha sido el aplicarle este controlador a un Quadcopter para que pueda aterrizar sobre una plataforma móvil. Es decir, con un 'Set-Point' dinámico. Esto se puede conseguir mediante la lectura de un April-Tag a través un algoritmo de percepción, calcular la distancia hasta el punto, e ir actualizando el punto final en cada instante, a medida que se aproxima, por lo tanto se podría trabajar en ello.

Otro de los puntos más interesantes de desarrollos futuros, es la ampliación del controlador también para regular la posición del Yaw en la aproximación. Con esto se podría lograr aterrizar el Drone en una posición determinada sobre la plataforma.

Existen en la actualidad otro tipo de control, que se llama Borroso. Es un ámbito muy atractivo de trabajo, ya que si se le añade un control PID a la salida de un controlador borroso sería capaz de modificar los parámetros de las ganancias del PID automáticamente. Lo cual le haría adaptarse mejor a los agentes externos como pueden ser las ráfagas de viento. Además se podría trabajar en un futuro con algoritmos evolutivos como puede ser el DE, el cual se utiliza para optimizar los parámetros de un controlador PID, para regular la estabilidad del movimiento del UAV (Roll, Pitch, Yaw) [12].

Capítulo 7

VII. PLANIFICACIÓN Y PRESUPUESTO

Planificación

En la siguiente tabla se muestran las horas aproximadas que se han invertido en el proyecto, el trabajo principal se realizó durando los meses Febrero, Marzo, Abril y Mayo. El número total de horas es de 400.

Tabla VII.1 Horas totales invertidas en el proyecto.

Mes	Horas/Semana	Horas/Mes
Noviembre	4	16
Diciembre	4	16
Enero	7	28
Febrero	10	40
Marzo	20	80
Abril	20	80
Mayo	20	80
Junio	15	60
Total		400

Presupuesto

El presupuesto se dividirá en costes materiales y costes humanos:

a) Costes Materiales

Para realizar el coste de los materiales que se han utilizado, tendremos en cuenta su precio inicial, periodo de amortización y periodo de ejecución del proyecto. La ecuación siguiente es la que se ha utilizado para calcular los costes materiales totales:

$$Coste = \frac{precio}{periodo\ amortización} \cdot Meses\ de\ uso$$

Tabla VII.2 Costes materiales del proyecto**.**

Producto	Precio (€)	Periodo de Amortización (Meses)	Periodo de uso (Meses)	Coste (€)
PC HP envy 17	800	24	8	266,67
Framework ROS	0	24	8	0,00
Licencia Microsoft Office	40	24	8	13,33
Gazebo Ardupilot	0	24	8	0,00
Ubuntu 16.04	0	24	8	0,00
Licencia MATLAB estudiante	0	24	8	0,00
Total				**280**

b) Costes Humanos

Según el convenio colectivo de ámbito estatal para ámbitos de educación universitaria e investigación [33]. Un investigador Universitario cobra mensualmente 1581,59 €.

Tabla VII.3 Costes Humanos del Proyecto.

Personal	Sueldo Mensual (€)	Meses
Investigador Universitario	1581,91	8
Total	12655,28	

Teniendo en cuenta los costes materiales y humanos tenemos un coste total del proyecto de:

Tabla VII.4 Costes totales del proyecto.

Tipo de Coste	Coste (€)
Materiales	280
Humanos	12655,28
Totales	12935,28

$$\mathbf{12935,28}\ € \ \boldsymbol{en}\ \mathbf{8}\ \boldsymbol{meses\ de\ ejecución}.$$

REFERENCIAS

[1] *Juhi Ajimera, Siddhartham Pr,Ramaravind K.M, Gautham Vasan, Naresh Balaji, V. Shankaranarayan "Autonomous Visual Tracking and Landing of a Quadrotor on a Moving Platform".* Third International Conference on Image information Processing, pp. 1-5, 2015.

[2] *Khaled A. Ghamry, Yinquin Dong, Mohamed A. Kamel, And Youmin Zhanghi "Real-Time Autonomous Take Off, Tracking And Landing Of UAV On A Moving UGV Platform".* 21th Meditherranean Conference on Control and Automation (MED), pp. 1238-1239, June 21-24 2016 Athens.

[3] *T.K. Venugopalan, Tawfig Taher, Prof. George Barbasthis*. "*Autonomous Landing of an aerial vehicle on an autonomous marine vehicle*.", pp. 4-6, Singapore 2012.

[4] *Miguel A. Olivares-Méndez, Iván F. Mondragón, Pascual Campoy And Carol Martínez.* **"***Fuzzy Controller for UAV-Landing Task Using 3D-Position Visual Estimation." pp. 1, 3-5,* Procceding of the 8th International FLINS Conference Madrid, Spain 2008.

[5] *Tiago Gomes Carreira*. "*Quadcopte Automatic Landing On A Docking Station."* Instituto Superior técnico, pp. 1-6, Lisboa Octubre 2013.

[6] *Lingyun Xu and Haibo Luo.* "*Towards Autonomous Tracking and Landing on Moving Target*." Conference on Real Time Computing and Robotics, pp. 620-623, June 6-9, 2016 Angkor wat, Cambodia.

[7] *David H. Warren and Edward R.Strelow. Electronic Spatial Sensing for the Blind: Contributions from Perception* 1985.

[8] *Hadrien Beck, Julien Leseueur Guillaume Charland- Arcand, Oauassima Akhrif, Samuel Gagné, François Gagnon and Denis Couillard. "Autonomous Takeoff And Landing Of A Quadcopter*." pp. 475-476, 2016 International Conference On ICUAS June 7-10 2016.

[9] *Pandi Li, Xim Chen, Chuntao Li.* **"***Emergency landing control technology for UAV".* Navigation and Control Conference. Yantai, China August 8-10 2014.

[10] *Paul_Lamb*. *"Dead Reckoning".* Recuperado de: *https://discourse.numenta.org/t/tracking-position-with-dead-reckoning/982.* El día 1-Abril-2017.

[11] *Samer Abdelmoeti. Should I use a PD or a PID for a position control application?* –Quora .Msc. student in System and Control at Twente University. Recuperado de: *https://www.quora.com/Should-I-use-a-PD-or-a-PID-for-a-position-control-application. El día 2-Abril-2017.*

[12] *Mohamed Reyad, M.Arafa, Elsayed A. Sallam.* "*An Optimal PID Controller for a Quadrotor System Based On DE Algorithm".* Tanta University. Egypt.

[13] *Ahmed Radwan Ibrahim*. *"Desing And Test A Control System For A Quadcopter*." Bachelor Thesis. German University In Cairo. 31 July, 2016.

[14] *Pasquale Longobardi*. "*Reactive Control System Desing and Implementation For Unmanned Aerial Vehicles*." Master Project report. March 15, 2017.

[15] *Amazon Prime Air Website*. Recuperado de: *https://www.amazon.com/Amazon-Prime-Air/b?node=8037720011, el día 2-Abril-2017.*

[16] *Adrian Weckler.* "Pony Express Couriers Make 'Ireland's First Drone Delivery' In Two Minutes Trip." Independent.ie newspaper.

[17] *Michelle Hennessy. "Ireland's First Legit Drone Drop Landed on a Boat at the Weekend."* Thejournal.ie. Recuperado de: *http://www.thejournal.ie/drone-delivery-3213381-Jan2017/.*

[18] *S. Bouabdallah.* "*Design and control of quadrotors with application to autonomous flying*." Doctoral dissertation, Ecole Polytechnique Federale de Lausanne pp. 1-129 2007.

[19] *Teppo Luukkonen*. "*Modellling and Control of Quadcopter."* Alto University. Espoo, August 22, 2011.

[20] *Chini, María Rita.* "*El efecto Giroscópico*." Colegio Konrad Lorenz, Luján de Cuyo, Mendoza.

[21] *Imagen Recuperada De*: *http: //www.robotshop.com/en/550mm-rtf-quadcopter-uav.html, el día 15-Abril-2017.*

[22] *Tianqu Zhao, Hong Jiang. "Landing System for A.R.Drone 2.0 using onboard camera and ROS."* Navigation and Control Conference, pp. 1101-1102, August 12-14 2016 Niajing China.

[23] *Jawhar Ghomman and Maarouf Saad. "Autonomous Landing of a Quadrotor on a Moving Platform." pp. 4-6,* April 2015.

[24] Controladora Pixhawk Autopilot PX4 e imágenes. *Datos recuperados de: https://pixhawk.org/modules/pixhawk, el día 24 –Abril-2017.*

[25] ROS. About ROS. *Información recuperada de: http://www.ros.org/about-ros/. El día 24-Abril-2017.*

[26] *"The main principles of ROS". Información recuperada de: http://www.generationrobots.com/blog/en/2016/03/ros-robot-operating-system-2/. día 24-Abril-2017.*

[27] *ErleRobotics, "Explicando MAVROS". Información recuperada de: http://erlerobotics.com/blog/ros-mavros-es/ el día 24-Abril-2017.*

[28] *WordPress.Org. Información recuperada de: https://codex.wordpress.org/Plugin_API, el día24-Abril-2017.*

[29] *PX4 Offboard Control Using MAVROS on ROS. 404warehouse. Imagen recuperada de: https://404warehouse.net/2015/12/20/autopilot-offboard-control-using-mavros-package-on-ros/. Recuperado el día 24-Abril-2017.*

[30] *QGROUNDCONTROL. MAVLink Micro Air Vehicle Communication Protocol. Información recuperada de: http://qgroundcontrol.org/mavlink/start. el día 24-Abril-2017.*

[31] *DJI. Uhmanned Aerial Vehicles Manufacturer. Imagen recuperada de: http://www.dji.com/. el día 02/06/2017.*

[32] F. Rinaldi, A. Gargioli, y F. Quagliotti, '*PID and Lq regulation of a multirotor attitude: Mathematical modelling, simulations and experimental results*,' '*Journal of Intelligent & Robotic Systems*', pp. 1-18, 2014.

[33] BOE '*Convenio colectivo de ámbito estatal para centros de educación universitaria e investigación'*. Recuperado de: *https://www.boe.es/boe/dias/2015/02/11/pdfs/BOE-A-2015-1343.pdf*. *El día 14/06/*2017.